We also give a comparison of the two techniques for error estimates in the numerical solution of stiff ordinary differential equations: the embedded method and the Richardson extrapolation scheme. The usual assumption that the Richardson extrapolation scheme is much worse than the embedding technique is not true. The basic method used to compare the two techniques is the ROW method.

A modified Richardson extrapolation scheme is proposed to reduce both the number of LU decompositions and the number of function evaluations by using $\gamma/2$ instead of γ in a Rosenbrock method for the step with 2h. The details of some fairly exhaustive numerical experiments are given.

ACKNOWLEDGEMENT

The author gratefully acknowledges the assistance of Dr. T. D. Bui for his patience and guidance in directing the project, and in reviewing and correcting this report. This project was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Department of Education of Quebec(FCAC). The author would also like to thank Professor G. Wanner(Geneve), Dr. P. Kaps(Innsbruck) and Dr. E. J. Doedel for many useful suggestions.

TABLE OF CONTENTS

Chapter 1

Introduction

It seems clear that many areas of engineering and scientific analysis require methods for solving sets of initial value ordinary differential equations (ODE's). The advent of the computer has significantly increased our ability to carry out the numerical solution of such equations. In the present thesis, we shall be concerned with the approximate numerical integration of systems of first order ordinary differential equations of the form

$$\frac{dy}{dx} = f(y), \qquad y(x_0) = y_0. \tag{1.1}$$

Most numerical methods for solving such equations are known as discrete variable methods. A sequence of discrete points $x_0, x_1, x_2, \ldots$ is generated, possibly with variable spacing, $h_n = x_{n+1} - x_n$. At each point x_n, the solution $y(x_n)$ is approximated by y_n which is computed from earlier values. A discrete variable method which provides a rule for computing y_{n+1} using k earlier values $y_n, y_{n-1}, \ldots, y_{n-k+1}$ is called a k-step method. If k=1, it is a single-step method (e.g. Runge-Kutta methods), and if k>1, it is a multistep method (e.g. Adams methods).

1.10. Stiff Initial-Value Problems; Stiffness Ratio

Many fields of application, notably chemical engineering and control theory, yield initial value problems involving systems of ordinary differential equations which exhibit a phenomenon which has come to be known as 'stiffness'. Generally speaking, a system of differential equations is said to be stiff if it has greatly differing time constants. 'Time constant' is a term used by engineers and physicists to refer to the rate of decay. For example, the equation $dy/dt = \lambda y$ has the solution $y = ce^{\lambda t}$. If λ is negative, then y decays by a factor of $1/e$ in time $T=1/\lambda$. T is called the time constant.

The problem of stiffness has been known for some time [13], but has in recent years attracted a great deal of attention. Recent surveys of methods for stiff problems produced by Watts[40], Bui[5] and Bjurel[3] report on many methods which have been proposed.

To determine whether or not an initial-value problem is stiff, consider the following example(Lambert[31]):

$$(1.2) \qquad y' = Ay, \qquad y(0)=[1,0,-1]^T$$

where

$$A = \begin{bmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{bmatrix} .$$

The solution is

$$y_1(x) = e^{-2x} + e^{-40x}(\cos 40x + \sin 40x)$$

$$y_2(x) = e^{-2x} - e^{-40x}(\cos 40x + \sin 40x)$$

$$y_3(x) = -e^{-40x}(\cos 40x - \sin 40x).$$

The graphs of $y_1(x)$, $y_2(x)$ and $y_3(x)$ are shown as continuous lines in Fig 1.1. The Jacobian $\partial f/\partial y$ is the constant matrix A whose eigenvalues λ are -2 and $-40\pm40i$. The solution y_1, y_2 and y_3 each has a rapidly decaying component, corresponding to $\lambda=-40\pm40i$. After a brief initial phase of the solution, two of the components, y_1 and y_2 are practically identical and vary quite slowly with x, while the third component, y_3, is virtually zero. We would like to proceed in a numerical solution with a larger step length h. However, for a stable numerical solution, most methods require that $|h\lambda_i|$, i=1,2,3 be bounded by a small number, typically of order 1 to 10. For example, for Euler's method it is necessary that $|h\lambda_i|<2$, thus we must satisfy $h<0.025$. Although the eigenvalues responsible for this severe restriction on h are $40\pm40i$, their contribution to the theoretical solutions are of no practical interest. The criterion of absolute stability forces us to use an extremely small value of h over the entire range of integration. As a result, the computation time necessary to integrate such systems can become excessive. This,

then, is the problem of stiff equations.

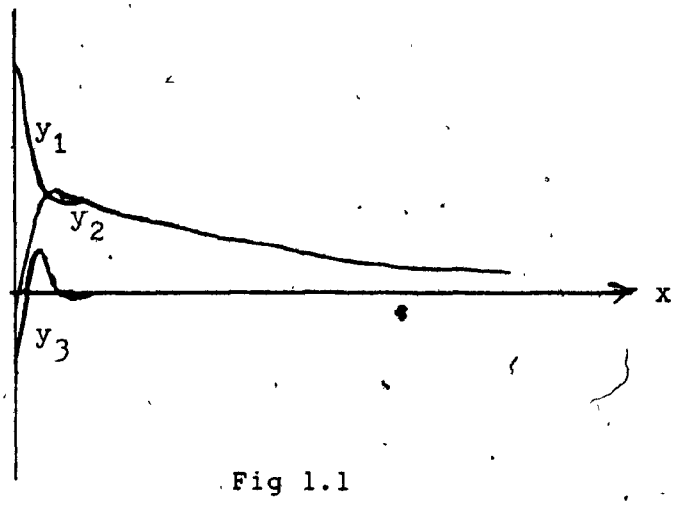


Fig 1.1

Definition 1.1 (Lambert [31])

The initial-value problem

$$y' = f(y), \quad y(a) = y_0, \quad x \in [a,b]$$

is said to be stiff in an interval $I \subseteq [a,b]$ if, for $x \in I$

(1) $Re(\lambda_i) < 0 \quad (i=1,2,\ldots,s)$; and

(2) $S(x) = \dfrac{\max\limits_{i=1,s} |Re(-\lambda_i)|}{\min\limits_{i=1,s} |Re(-\lambda_i)|} >> 1$

where the λ_i are the eigenvalues of $\partial f/\partial y$ evaluated on the solution $y(x)$ at x.

The ratio $S(x)$ may be termed the (local) "Stiffness Ratio" of the problem. Problems may be considered to be marginally stiff if $S(x)$ is $O(10)$, while stiffness ratios up to $O(10^6)$ - $O(10^{10})$ and higher are not uncommon in practical problems.

1.20 Stability For Stiff Problems

The concept of absolute stability of a numerical method imposes the constraint that an error introduced at some step not to be magnified over subsequent steps. To examine absolute stability of a method, we consider the test equation $y'=\lambda y$, where λ is a complex number whose real part is negative. Then an absolute stability region is defined as the set of points $h\lambda$ in the plane such that when the method is applied to this test equation, $|y_{n+1}| \leq |y_n|$. For the absolute stability of the numerical solution to a stiff system, it is necessary to use a step size h such that every one of the values $z_i := h\lambda_i$ lies within the region of absolute stability of the numerical method. For a method with a finite absolute stability region the step size is thus restricted to the order of magnitude of the smallest time constant of the system. If a problem is very stiff, we are forced into the highly undesirable computational situation of having to integrate numerically over a long range, using a steplength which is everywhere excessively small relative to the interval.

The problem, then, is to develop methods that do not restrict the step size for stability reasons, i.e. methods that possess regions of absolute stability that extend to infinity in the half-plane $Re(h\lambda) < 0$. Several definitions, which call for the method to possess some

'adequate' region of absolute stability, have been proposed.

Definition 1.2 (Dahlquist[14]) : A numerical method is said to be A-stable if its region of absolute stability contains the whole of the left-hand half-plane Re $(h\lambda) < 0$ (Fig 1.2).

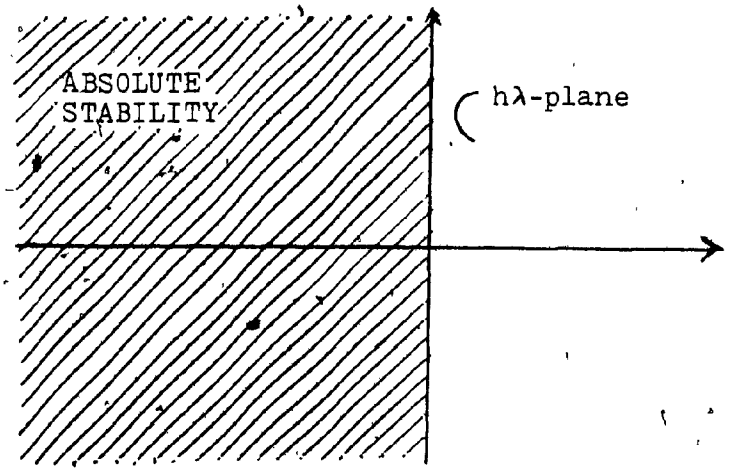


Fig 1.2 A-Stability

If an A-stable method is applied to a stiff system, then the difficulties described above will disappear. However, A-stability is a severe requirement to ask of a numerical method, as Dahlquist[14] proved that an explicit linear multistep method can not be A-stable, and that the order of an A-stable implicit linear multistep method could not exceed two, the implicit trapezoidal rule being the most accurate of such methods. In view of this, several less demanding stability definitions have been proposed.

Definition 1.3 (Widlund[41]) : A numerical method is said to be A(α)-stable, $\alpha \in (0, \pi/2)$, if its region of absolute stability contains the infinite wedge $|\arg(-\lambda)| < \alpha$ (Fig 1.3). A method is said to be A(0)-stable if it is A(α)-stable for some sufficiently small $\alpha \in (0, \pi/2)$.

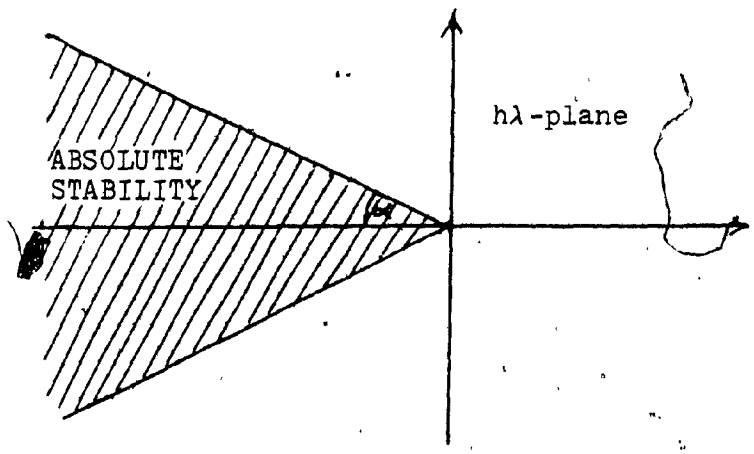


Fig 1.3 A(α)-Stability

The above definitions are concerned only with stability, whereas Gear[19] defines a more complex property, involving both stability and accuracy of approximations to the exponential eigensolutions.

Definition 1.4

A method is stiffly stable if in the region R_1 ($Re(h\lambda) \leq D$) it is absolutely stable, and in R_2 ($D < Re(h\lambda) < \alpha$, $|Im(h\lambda)| < \theta$) it is accurate. (Fig 1.4)

The regions R_1 and R_2 of the complex $h\lambda$-plane are

shown in Fig 1.4. The reasoning behind this definition is as follows. $e^{h\lambda}$ is the change in a component in one step due to an eigenvalue λ. If $h\lambda=u+iv$, then the change in magnitude is e^{u}. Those eigenvalues which represent rapidly decaying components in the transient solutions will correspond to the values of $h\lambda$ in R_1. We are not interested in the accuracy of the components that are very small, so for some D, $u<D<0$, we are willing to ignore all components in R_1. We just require that the method be absolutely stable. The remaining eigenvalues of the system represent terms in the solution which we would like to represent accurately as well as stably; by suitable choice of h, $h\lambda \in R_2$ ensures an accurate approximation.

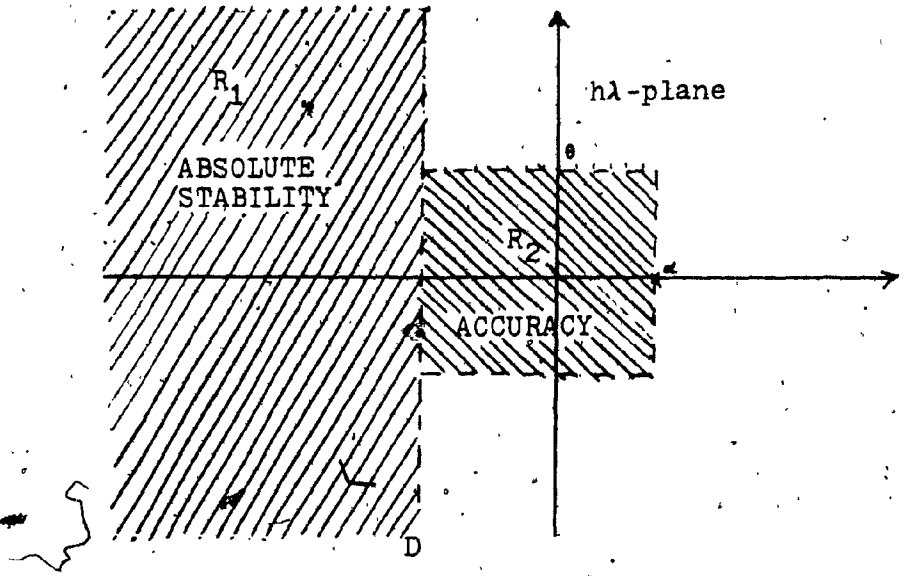


Fig 1.4 Stiff Stability

In some respects the A-stability property is not stringent enough, and studies of A-stable one-step methods

have produced a number of additional stability concepts. Axelsson[2] has defined methods to be L-stable (or maximally damped at infinity) if for the equation $y'=\lambda y$, $y_{n+1}/y_n \to 0$ as $Re(h\lambda) \to -\infty$. Such methods would cause rapidly decaying components to also decay rapidly in the numerical approximation, so that it would not be necessary to use small steps even when these components were still present - at least in linear systems.

Definition 1.5(Ehle[15])

A one-step numerical method is said to be L-stable if it is A-stable and, in addition, when applied to the scalar test equation $y'=\lambda y$, λ is a complex constant with $Re\lambda<0$, it yields $y_{n+1} = Q(h\lambda)y_n$, where $|Q(h\lambda)| \to 0$ as $Re(h\lambda) \to -\infty$.

Chapter 2.

Semi-Implicit Runge-Kutta Methods

2.10 Runge-Kutta Methods

The general v-stage Runge-Kutta method with constant coefficients for the numerical solution of the initial value problem (1.1) is given by

$$k_i = f(y_n + h \sum_{j=1}^{v} a_{ij} k_j), \quad i=1,\ldots,v.$$

$$y_{n+1} = y_n + h \sum_{i=1}^{v} b_i k_i \qquad (2.1)$$

where a_{ij}, b_i are real coefficients, h denotes the step size and $y_n = y(t_n)$. The matrix

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1v} & 0 \\ \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdots & \cdot & \cdot \\ a_{v1} & \cdots & a_{vv} & 0 \\ b_1 & \cdots & b_v & 0 \end{pmatrix}$$

is called the coefficient matrix for the method and the Runge-Kutta method is called explicit if

$$a_{ij} = 0 \quad \text{for } j \geq i. \qquad (2.2)$$

If A is such that

$$a_{ij} = 0 \quad \text{for } j > i \qquad (2.3)$$

the method is called semi-implicit (or semi-explicit). When (2.3) does not hold we have an implicit method. There have been several investigations [15], [12] of semi-implicit methods since Butcher[9] proposed them in 1964. But no systematic study was carried out until the work of Norsett[33].

2.11 Rosenbrock Procedures

Rosenbrock[35] proposed an extension of the Runge-Kutta process by introducing the Jacobian directly into the coefficients. The most general form is

$$k_i = f(y_n + h\sum_{j=1}^{i-1} a_{ij}k_j) + \gamma f'(y_n + \sum_{j=1}^{i-1} c_{ij}k_j)k_i \qquad i=1,\ldots,v$$

(2.4)

$$y_{n+1} = y_n + h\sum_{i=1}^{v} b_i k_i \,.$$

They can be regarded either as modifications of explicit Runge-Kutta methods or as linearizations of semi-implicit Runge-Kutta methods. Whereas a semi-implicit Runge-Kutta method would call for the solution of v sets of non-linear equations, the methods (2.4) call only for the solution of v sets of linear equations.

2.12 Modified Rosenbrock Methods (ROW Methods)

In order to obtain better stability properties of the

Rosenbrock methods, Wanner[38] and Wolfbrandt[42] introduced the coefficients γ_{ij} into (2.4) to obtain the modified Rosenbrock methods (Rosenbrock-Wanner methods, or briefly ROW-methods). The v-stage ROW method is defined by

$$k_i = f(y_n + h\sum_{j=1}^{i-1} a_{ij}k_j) + h\sum_{j=1}^{i} \gamma_{ij} f'(y_n)k_j \qquad i=1,\ldots,v$$

(2.5)

$$y_{n+1} = y_n + h\sum_{i=1}^{v} b_i k_i \quad .$$

Letting $\gamma_{ii} = \gamma$, we can rewrite (2.5) as

$$(I-\gamma hf'(y_n))k_i = hf(y_n + \sum_{j=1}^{i-1} a_{ij}k_j) + hf'(y_n)\sum_{j=1}^{i-1} \gamma_{ij}k_j \quad ,$$

(2.6)

$$y_{n+1} = y_n + \sum_{i=1}^{v} b_i k_i \quad .$$

For $\gamma_{ij}=0$ the ROW methods reduce to the usual Runge-Kutta (Rosenbrock) methods. Therefore ROW-methods can be considered as generalized Runge-Kutta methods, which can be obtained - roughly speaking - by 1 Newton iteration of the diagonally implicit Runge-Kutta method.

2.20 Derivation of Order Conditions

In this section, we will give a short review of the work done by Butcher [8] and Hairer and Wanner [23], [24], and then we will present the derivation of the order conditions for ROW methods.

2.21 Monotonically Labelled Trees and Trees

Definition 2.1 (Monotonically Labelled (rooted) Trees, LT) Let n be a nonnegative integer and $t:\{2,\ldots,n\} \to \{1,\ldots,n\}$ be a mapping which satisfies

$$t(i) < i \text{ for } i = 2,\ldots,n.$$

Then we call t a monotonically labelled tree of order n. The order is denoted by $\rho(t)$.

Example. Fig 2.1 gives all LT's up to order 3.

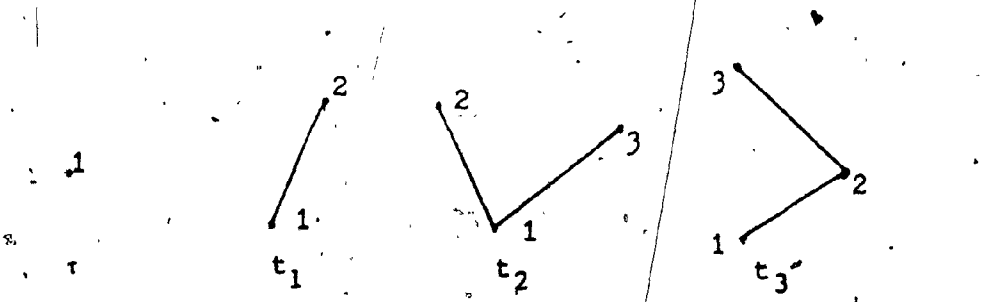


Fig 2.1 LT's

The node with label 1 is called the root.

Definition 2.2 (Tree, T). A tree is an equivalence class of LT's which represents the same graph but differ in numeration. This equivalence relation can be defined by

$t = u$ <==> 1) $\rho(t) = \rho(u)$

2) There exists a permutation σ of $\{1,\ldots,\rho(t)\}$ such that $\sigma(1)=1$ and the arcs of u are the elements of $\{(\sigma(t(i)),\sigma(i)),\ i=2,\ldots,\rho(t)\}$.

Example. Fig 2.2 gives examples of trees. The notation t_1, t_2 is taken from Fig 2.1.

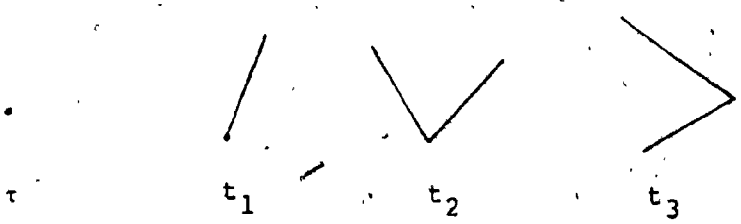


Fig 2.2 Trees

Definition 2.3 (Recursive Representation of Trees) Let $t_1,\ldots,t_m$ ($m \geq 0$) be monotonically labelled trees with $\rho(t_i) \geq 1$. We define a new LT, $t=[t_1,\ldots,t_m]$ of order $1 + \rho(t_1)+\ldots+ \rho(t_m)$. With

$$s_i = \sum_{j=1}^{i-1} \rho(t_j)$$

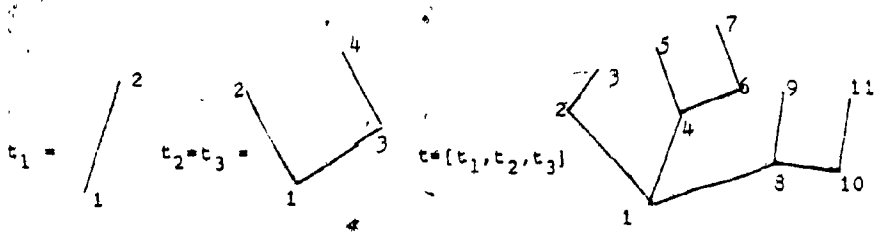
define

$$t(2+s_i) := 1 \quad \text{for } i=1,\ldots,m$$

and

$$t(1+s_i+j) := 1 + s_i + t_i(j) \quad \text{for } i=1,\ldots,m \text{ and } j=2,\ldots,\rho(t_i).$$

Example Fig 2.3 shows the simplicity of this construction.



i	j	$t_i(j)$	s_i	$t(2+s_i)=1$	$t(1+s_i+j)=1+s_i+ti(j)$
1	2	1	0	$t(2)=1$	$t(3)=2$
2	2	1	2	$t(4)=1$	$t(5)=4$
	3	1			$t(6)=4$
	4	3			$t(7)=6$
3	2	1	6	$t(8)=1$	$t(9)=8$
	3	1			$t(10)=8$
	4	3			$t(11)=10$

Fig 2.3 Recursive Representation of LT.

Similarly, trees, the equivalence class of LT's, can also be represented recursively.

Definition 2.4 Every tree t with $\rho(t) \geq 1$ can be decribed as $t=[t_1,\ldots,t_m]$ where $m \geq 0$ and $\rho(t_i) \geq 1$. The notation

$$t = [\,t_1,\ldots,t_m\,]$$

indicates, that the trees $t_1,\ldots,t_m$ remain after the root of t and the adjacent arcs have been removed (Fig 2.4).

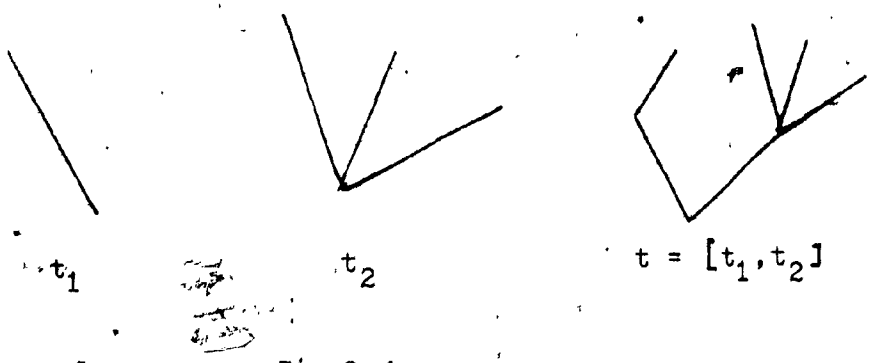


Fig 2.4

We denote the tree consisting of its root only by τ. We let T be the set of all trees and ST be the subset of all "Special trees". i.e. the trees τ, $[\tau]$, $[[\tau]]$,... which have no ramifications. We call a node of a tree t, where $t \in ST$, single-branched if there is exactly one upward branch leaving this node (see Fig 2.5).

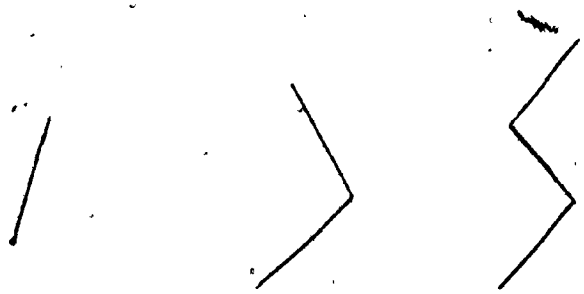
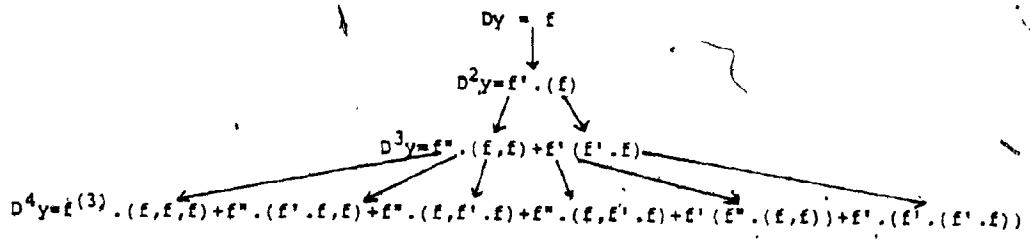


Fig 2.5 ST

Example. (Application of Trees) The differential operator operating on y

$$(2.7) \quad D = \frac{d}{dy}(\,).f(y)$$

can be interpreted by means of trees as follows.



. . . . . .

Here, for example, f".(f'.f,f) means that the bilinear map f" operates on the vectors f'.f and f. By representing each f by a node, each f' by a node with one branch, and each f" by a node with two branches, and so on, we can interpret this by

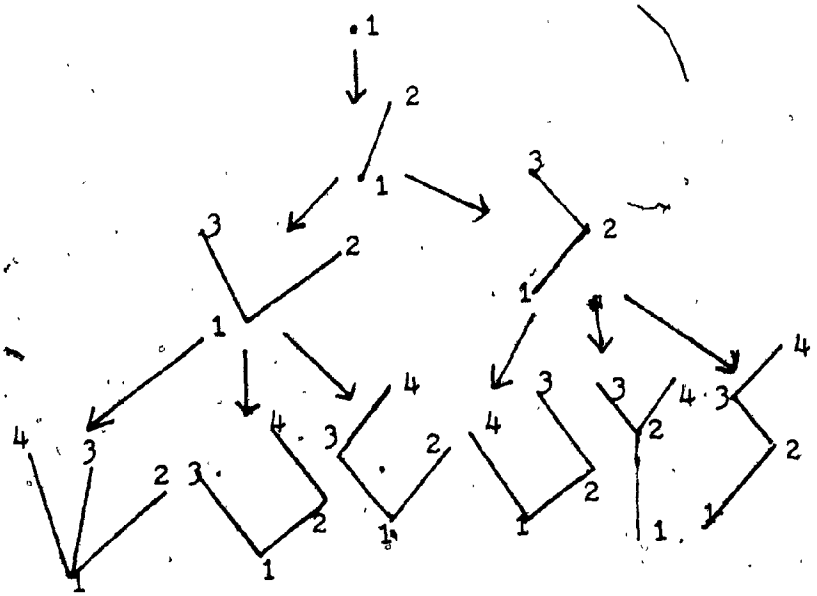


Fig 2.6

Definition 2.5 A tree u is a subtree of the tree t, if

1) $\rho(u) \leq \rho(t)$,

2) $t\lfloor\{2,\ldots,\rho(t)\} = u$.

Definition 2.6 A pair of tree uct is a tree t together with a subtree u of t.

We denote the set of all pairs of trees by PT. In the geometric representation, a subtree is distinguished by double lines and circle nodes (Fig 2.7)

Example

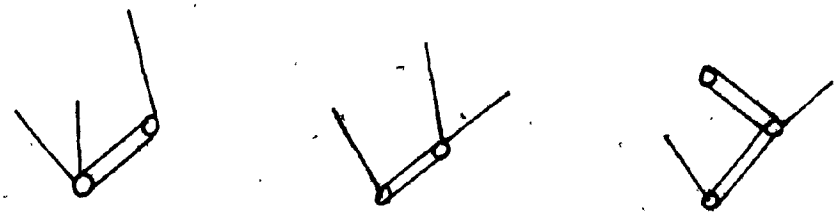


Fig 2.7 Pairs of Trees

2.22 Butcher Series

Definition 2.7 (Elementary Differentials)

Consider the stiff system of first-order ordinary differential equations (1.1), we define for every $t \in T$, a function $F(t)y: R \to R$, recursively by

$$F(\emptyset)(y) = y$$

(2.8)

$$F(t)(y) = f^m(y).(F(t_1)(y),\ldots,F(t_m)(y))$$

where $y \in R$, $t=[t_1,\ldots,t_m]$ and $f^{(m)}$ denotes the m-th derivative of f. We call the function F(t) an elementary differential.

Example

$$F(\tau)(y) = f(y)$$

$$F(t_1)(y) = f'(y).f(y)$$

$$F(t_2)(y) = f''(y).(f(y),f(y))$$

$$F(t_3)(y) = f'(y).f'(y).f(y)$$

where τ, t_1, t_2 and t_3 are defined as in Fig 2.2.

Definition 2.8 If $a : T \to R$ is a mapping, the series

$$(2.9) \qquad B(a,y_n) = \sum_{t\in T} a(t)F(t)(y_n)\frac{h^r}{r!} \qquad (r=\rho(t))$$

is called a Butcher series.

Example. The identity map $B(o,y_n)=y_n$ is a Butcher series with

$$o(t) = \begin{cases} 1 & t = \emptyset, \\ 0 & t \neq \emptyset. \end{cases}$$

Example hf(y) for an initial value problem $y'=f(y)$ is a Butcher series B(b,y) with

$$b(t) = \begin{cases} 1 & t = \tau, \\ 0 & t \neq \tau. \end{cases}$$

Example The solution of the differential equation $y'=f(y)$ can be written as $y(x_0+h) = \sum_{t\in LT} F(t)(y_0)\frac{h^r}{r!}$. This is a Butcher series $B(p,y_0)$ where

$$p(t) = 1 \text{ for all } t \in T .$$

Example The solution $y(x+kh)$ is a Butcher series $B(p_k,y_0)$ with $p_k(t) = k^{\rho(t)}$.

Theorem 2.1 Let $a : T \to R$, $b : T \to R$ and $a(\emptyset) = 1$. Then the composition of the two corresponding Butcher series

$$(2.10)\quad B(b,B(a,y_n)) = B(ab,y_n)$$

is again a Butcher series with

$$(2.11)\quad (ab)(t) = \sum_{uct\in PT} b(u)\binom{r}{i}\frac{\alpha(uct)}{\alpha(t)}a(uct)$$

and

$$a(uct) = \prod_{z\in d(uct)} a(z), \quad r=\rho(t),\ i=\rho(u) .$$

Here, as defined in [24], $\alpha(t)$ and $\alpha(uct)$ is the number of possibilities of monotonic labelling the nodes of t and uct respectively, and $d(uct)$ is the tuple of trees remaining after removing the subtree u. Examples and the Proof of Theorem 2.1 are given in [24].

2.23 Derivation of Order Conditions for ROW methods

In this section, we obtain the order conditions for ROW methods using the theories of trees and Butcher series presented above. Anyone who has ever tried to obtain the equations of conditions for Runge-Kutta methods by Power series expansion will appreciate the following simple procedures.

For the ROW methods defined in (2.5), we let

$$(2.12)\quad u_i := y_n + \sum_{j=1}^{i-1} a_{ij} k_j \quad .$$

We study the Power series of y_{n+1}, k_i and u_i about h, these turn out to be Butcher series.

Theorem 2.2 The functions of u_i, k_i and y_{n+1} of (2.5) are Butcher series defined by

$$u_i(h)=B(\underline{u}_i,y_n),\quad k_i(h)=B(\underline{k}_i,y_n),\quad y_{n+1}(h)=B(\underline{y}_{n+1},y_n)$$

where the coefficients $\underline{u}_i(t)$, $\underline{k}_i(t)$ and $\underline{y}_{n+1}(t)$ for $t\in T$ can be expressed recursively by

$$(2.13a)\quad \underline{u}_i(\emptyset)=1,\qquad \underline{u}_i(t)=\sum_{j=1}^{i-1} a_{ij}\underline{k}_j$$

$$(2.13b)\quad \underline{k}_i(\emptyset)=0,\qquad \underline{k}_i(\tau)=1$$

$$\underline{k}_i(t)=\rho(t)\underline{u}_i(t_1)\ldots\underline{u}_i(t_m)+\begin{cases} 0 \text{ if } t=[t_1,\ldots,t_m],\ m>1 \\ \rho(t)\sum_{j=1}^{i}\gamma_{ij}\underline{k}_j(t_1) \text{ if } t=[t_1] \end{cases}$$

$$(2.13c)\quad \underline{y}_{n+1}(\emptyset)=1,\qquad \underline{y}_{n+1}(t)=\sum_{i=1}^{\nu} b_i\underline{k}_i(t) \quad .$$

Proof. (2.13a) and (2.13c) are directly obtained from (2.5) and (2.12). As we see in section 2.22 and proved in [24], $hf(u_i)=B(\underline{u}_i', y_n)$ is a Butcher series with coefficients $\underline{u}_i'(t)=\rho(t)\underline{u}_i(t_1)\ldots\underline{u}_i(t_m)$. This gives the first term of (2.13b). Since $f'(y_n)F(t_1)(y_n) = F([t_1])(y_n)$,

$$hf'(y_n)k_j(h) = \sum_{t\in LT} \underline{k}_j(t)\, hf'(y_n)[F(t)(y_n)]\frac{h^{\rho(t)}}{\rho(t)!}$$

$$= \sum_{\substack{t\in LT \\ t\neq\emptyset}} \rho([t])\,\underline{k}_j(t)F([t])(y_n)\frac{h^{\rho([t])}}{\rho([t])!}.$$

which is a Butcher series with coefficients not equal to zero belonging to trees with a single-branched root. This yields the second term of (2.13b).

We can simplify formulas (2.13) by putting

$$a_{ij} := 0 \quad \text{for } j \geq i \,,$$

$$\gamma_{ij} := 0 \quad \text{for } j > i \,,$$

(2.14)

$$\beta'_{ij} := a_{ij} + \gamma_{ij} \quad \text{and} \quad \beta_{ij} := \begin{cases} \beta'_{ij} & i>j \,, \\ 0 & i=j \,, \end{cases}$$

and defining

$$\Gamma(\tau):=1, \qquad \Gamma(t):=\rho(t)\,\Gamma(t_1)\ldots\Gamma(t_m) \text{ for } t=[t_1,\ldots,t_m].$$

By subsitution, we obtain

Theorem 2.3 For every tree $t \in T$ and $t\neq\emptyset$ the coefficients of the Butcher series of y_{n+1} are given by

$$\underline{y}_{n+1}(t) = \Gamma(t)\sum_{i=1}^{\nu} b_i \underline{\gamma}_i(t)$$

where $\gamma_i(t)$ are defined recusively by

$$\gamma_i(\tau) := 1$$

$$\gamma_i(t) := \begin{cases} \sum_{j_1 \ldots j_m} a_{ij1} \ldots a_{ijm} \gamma_{j1}(t_1) \ldots \gamma_{jm}(t_m), & \text{for } t=[t_1,\ldots,t_m] \\ \sum_j \beta'_{ij} \gamma_j(t_1), & \text{for } t=[t_1] \end{cases} \tag{2.15}$$

Because of the recursive properties of $\gamma_i(t)$, they can be easily obtained from the shape of the tree. One has to attach to each node of t a summation letter starting with "i" at the root, then γ_i is equal to the sum over the product containing

β'_{jk} - whenever a single-branched node "j" is directly connected with an upper node "k";

a_{ij} - whenever a multiple-branched node "j" is directly connected with an upper node "k".

Example Consider the tree in Fig 2.8

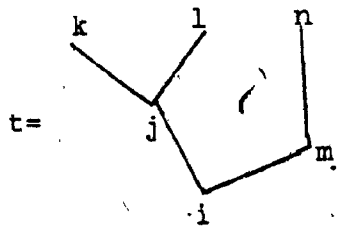


Fig 2.8

Then $y_{n+1}(t) = 6.3.2 \sum_i b_i \gamma_i(t)$

where $\gamma_i(t) = \sum_{j,k,l,m,n} a_{ij} a_{jk} a_{jl} a_{im} \beta'_{mn}$

The expression for $\gamma_i(t)$ can be simplified further by the elimination of the top nodes(k,l,n in this example) by

putting

$$(2.16)\quad \beta'_i := \sum_j \beta'_{ij}\,, \qquad \beta_i := \sum_j \beta_{ij}\,, \qquad a_i := \sum_j a_{ij}\,.$$

Then $\Psi_i(t) = \sum_{j,m} a_{ij} a_j^2 a_{im} \beta'_m$.

Table 2.1 gives the expressions of $\Psi_i(t)$ for all trees up to order 3.

Number	t	$\rho(t)$	$\Gamma(t)$	$\Sigma b_i \Psi_i$
1	•	1	1	Σb_i
2	/	2	2	$\Sigma b_i \beta'_i$
3	V	3	3	$\Sigma b_i a_i^2$
4	>	3	6	$\Sigma b_i \beta'_{ij} \beta'_j$

Table 2.1

As we see in section 2.2, the true solution $y(x_n+h)$ of the differential equation $y'=f(y)$ is a Butcher series $B(p,y_n)$ with coefficients $\rho(t)=1$ for all $t \in T$. Thus we have

Corollary 2.1 Method (2.5) is of order p iff

$$(2.17)\quad \Gamma(t) \sum_i b_i \Psi_i(t) = 1\,,$$

$$\text{i.e.} \qquad \sum_i b_i \Psi_i(t) = \frac{1}{\Gamma(t)} \qquad \text{for } \rho(t) \leq p\,,$$

and this equation is not true for at least one tree t of order p+1.

The principal truncation error $\varepsilon(t)$ is thus given by

$$(2.18)\quad \varepsilon(t) = \sum_{t \in LT,\, \rho(t)=p+1} \frac{h^{p+1}}{(p+1)!}\, e(t)\, F(t)(y_n)$$

where we call $e(t) = 1 - \gamma_{n+1}(t)$ the error coefficients.

2.24 Simplifying Assumptions

The equations of conditions given in the last section are nonlinear equations in the parameters a_i and a_{ij}. However, they can be simplified by the following two processes (Proposition 2.1 and 2.2).

Proposition 2.1

Let u_1, u_2, v_1, v_2 and v_3 be trees as sketched in Fig 2.9, where the encircled parts are assumed to be identical. Then the condition

$$(2.19) \qquad \sum_k a_{ik}\beta'_k = a_i^2/2 \qquad (i=2,\ldots,\nu)$$

$$\text{or} \qquad \sum_k a_{ik}\beta_k = a_i(a_i/2-\gamma)$$

implies that for y_{n+1} defined by (2.13c), $y_{n+1}(u_1) = y_{n+1}(u_2)$.

Similarly,

$$(2.20) \qquad \sum_i b_i\beta'_{ik} = b_k(1-a_k)$$

$$\text{or} \qquad \sum_i b_i\beta_{ik} = b_k(1-a_k-\gamma) \qquad (1 \leqslant k \leqslant \nu)$$

implies that $y_{n+1}(v_1)=1$ and $y_{n+1}(v_2)=1$ implies $y_{n+1}(v_3)=1$.

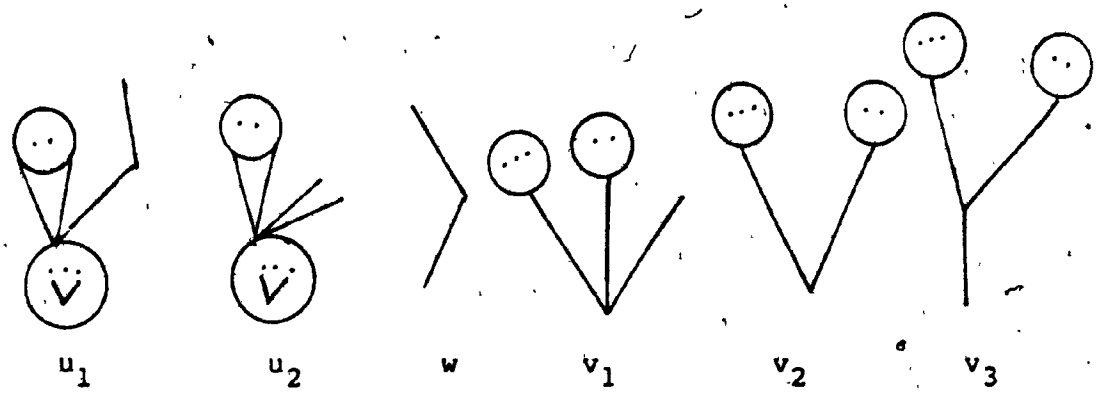


Fig 2.9

Proof. (direct verification) In the representation of u_1 and u_2 all trees are equal with one exception. But this can be written as

$$u_1 = [t_1,\ldots,t_k,t_{k+1},\ldots,t_m,w] \quad \text{with } w=[\tau]$$

$$u_2 = [t_1,\ldots,t_k,\tau,\tau,t_{k+1},\ldots,t_m]$$

and $\rho(u_1)=\rho(u_2)$. If we apply equation (2.19) to u_1 and u_2, we obtain (2.15) if $\underline{y}_{n+1}(u_1)=\underline{y}_{n+1}(u_2)$. The second part of this proposition can be proved in the same way.

Proposition 2.2

Let

$$\gamma_{ii} := \gamma := \beta'_{ii} \qquad (1 \leqslant i \leqslant v)$$

and move all terms that contain γ to the right hand side of (2.17). This gives a polynomial $P_k(\gamma)$ in γ (k denotes the tree number), and the number of terms in Ψ_i is considerably reduced. Also we define $\phi_i(t)$ exactly as $\Psi_i(t)$ in (2.15) except that β'_{ij} is replaced by β_{ij}.

Further for $t \in T$ and a positive integer j, we define

$$V(t,j) = \{s \in T,\ s \text{ is obtained from } t \text{ by removing } j \text{ single branched nodes}\}.$$

If t has less than j single-branched roots, then $V(t,j)=\emptyset$, and we denote

$$N(t,s) = \text{number of possibities to obtain } s \text{ by removing } j \text{ single-branched nodes from } t.$$

Theorem 2.4 If the order conditions (2.17) are satisfied for all trees of order $\leq p(t)-1$, then

$$(2.21) \qquad \sum_i b_i \phi_i(t) = \sum_{j\geq 0} (-\gamma)^j \sum_{s\in V(t,j)} N(t,s)/\Gamma(s)$$

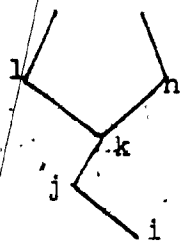
is equivalent to (2.17) for t.

Proof. If we replace $\phi_i(t)$ by (2.21), $\beta_{ij}=(\beta'_{ij}-\gamma\delta_{ij})$ and multiply the powers of γ, we obtain

$$\sum_i b_i \phi_i(t) = \sum_{j\geq 0} (-\gamma)^j \sum_{s\in V(t,j)} N(t,s) \sum_i b_i \Psi_i(s) .$$

If we now replace $\sum_i b_i \Psi_i(s)$ by $1/\Gamma(s)$, we obtain (2.21).

Example



j	1	1	2	2	2	3	3	4
s								
N(t,s)	2	2	1	1	4	2	2	1
$\Gamma(s)$	6.5.2.2	6.5.4.2	5.4.3	5.2.2	5.4.2	4.3	4.2	3

Thus the corresponding order condition is

$$\sum_{i,j,k,l,n} b_i \beta_{ij} \beta_{jk} a_{kl} \beta_l a_{kn} \beta_n = \frac{1}{7.6.5.2.2} - \frac{\gamma}{40} + \frac{\gamma^2}{6} - \frac{5\gamma^3}{12} + \frac{\gamma^4}{3} .$$

Table 2.2 lists the order conditions for ROW methods up to order 6.

number	tree	order	
1	•	1	$\Sigma b_i = P_1$
2		2	$\Sigma b_i \beta_i = P_2$
3		3	$\Sigma b_i a_i^2 = P_3$
4		3	$\Sigma b_i \beta_{ij} \beta_j = P_4$
5		4	$\Sigma b_i a_i^3 = P_5$
6		4	$\Sigma b_i a_i a_{ik} \beta_k = P_6$
7		4	$\Sigma b_i \beta_{ik} a_k^2 = P_7$
8		4	$\Sigma b_i \beta_{ik} \beta_{kl} \beta_l = P_8$
9		5	$\Sigma b_i a_i^4 = P_9$
10		5	$\Sigma b_i a_i^2 a_{ik} \beta_k = P_{10}$
11		5	$\Sigma b_i a_{ik} \beta_k a_{il} \beta_l = P_{11}$

12		5	$\Sigma b_i a_i a_{ik} a_k^2 = P_{12}$
13		5	$\Sigma b_i a_i a_{ik} \beta_{kl} \beta_l = P_{13}$
14		5	$\Sigma b_i \beta_{ik} a_k^3 = P_{14}$
15		5	$\Sigma b_i \beta_{ik} a_k a_{kl} \beta_l = P_{15}$
16		5	$\Sigma b_i \beta_{ik} \beta_{kl} a_l^2 = P_{16}$
17		5	$\Sigma b_i \beta_{ik} \beta_{kl} \beta_{lm} \beta_m = P_{17}$
18		6	$\Sigma b_i a_i^5 = P_{18}$
19		6	$\Sigma b_i a_i^3 a_{ik} \beta_k = P_{19}$
20		6	$\Sigma b_i a_i a_{ik} \beta_k a_{il} \beta_l = P_{20}$
21		6	$\Sigma b_i a_i^2 a_{ik} a_k^2 = P_{21}$
22		6	$\Sigma b_i a_i^2 a_{ik} \beta_{kl} \beta_l = P_{22}$
23		6	$\Sigma b_i a_{ik} \beta_k a_{il} a_l^2 = P_{23}$
24		6	$\Sigma b_i a_{ik} \beta_k a_{il} \beta_{lm} \beta_m = P_{24}$
25		6	$\Sigma b_i a_i a_{ik} a_k^3 = P_{25}$

26		6	$\Sigma b_i a_i a_{ik} a_k a_{kl} \beta_l = P_{26}$
27		6	$\Sigma b_i a_i a_{ik} \beta_{kl} a_l^2 = P_{27}$
28		6	$\Sigma b_i a_i a_{ik} \beta_{kl} \beta_{lm} \beta_m = P_{28}$
29		6	$\Sigma b_i \beta_{ik} a_k^4 = P_{29}$
30		6	$\Sigma b_i \beta_{ik} a_k^2 a_{kl} \beta_l = P_{30}$
31		6	$\Sigma b_i \beta_{ik} a_{kl} \beta_l a_{km} \beta_m = P_{31}$
32		6	$\Sigma b_i \beta_{ik} a_k a_{kl} a_l^2 = P_{32}$
33		6	$\Sigma b_i \beta_{ik} a_k a_{kl} \beta_{lm} \beta_m = P_{33}$
34		6	$\Sigma b_i \beta_{ik} \beta_{kl} a_l^3 = P_{34}$
35		6	$\Sigma b_i \beta_{ik} \beta_{kl} a_l a_{lm} \beta_m = P_{35}$
36		6	$\Sigma b_i \beta_{ik} \beta_{kl} \beta_{lm} a_m^2 = P_{36}$
37		6	$\Sigma b_i \beta_{ik} \beta_{kl} \beta_{lm} \beta_{mn} \beta_n = P_{37}$

The trees numbered 19, 20, 23, 24, 26, 29, 30, 31, 32, 33 and 35 can be eliminated by the simplying

assumption(proposition 2.1). Thus we have

$P_1 = 1$ $\qquad$ $P_2 = \frac{1}{2} - r$

$P_3 = \frac{1}{3}$ $\qquad$ $P_4 = \frac{1}{6} - r + r^2$

$P_5 = \frac{1}{4}$ $\qquad$ $P_6 = \frac{1}{8} - \frac{r}{3}$

$P_7 = \frac{1}{12} - \frac{r}{3}$ $\qquad$ $P_8 = \frac{1}{24} - \frac{r}{2} + \frac{3r^2}{2} - r^3$

$P_9 = \frac{1}{5}$ $\qquad$ $P_{10} = \frac{1}{10} - \frac{r}{4}$

$P_{11} = \frac{1}{20} - \frac{r}{4} + \frac{r^2}{3}$ $\qquad$ $P_{12} = \frac{1}{15}$

$P_{13} = \frac{1}{30} - \frac{r}{4} + \frac{r^2}{3}$ $\qquad$ $P_{14} = \frac{1}{20} - \frac{r}{4}$

$P_{15} = \frac{1}{40} - \frac{5}{24} + \frac{r^2}{3}$ $\qquad$ $P_{16} = \frac{1}{60} - \frac{r}{6} + \frac{r^2}{3}$

$P_{17} = \frac{1}{120} - \frac{r}{6} + r^2 - 2r^3 + r^4$ $\qquad$ $P_{18} = \frac{1}{6}$

$P_{21} = \frac{1}{18}$ $\qquad$ $P_{22} = \frac{1}{36} - \frac{r}{5} + \frac{r^2}{3}$

$P_{25} = \frac{1}{24}$ $\qquad$ $P_{27} = \frac{1}{72} - \frac{r}{15}$

$P_{28} = \frac{1}{144} - \frac{r}{10} + \frac{3r^2}{8} - \frac{r^3}{3}$ $\qquad$ $P_{34} = \frac{1}{120} - \frac{r}{10} + \frac{r^2}{4}$

$P_{36} = \frac{1}{360} - \frac{r}{20} + \frac{r^2}{4} - \frac{r^3}{3}$ $\qquad$ $P_{37} = \frac{1}{720} - \frac{r}{24} + \frac{5r^2}{2} - \frac{5r^3}{3} + \frac{5r^4}{2} - r^5$

2.30 Numerical Example

We list the order conditions for a 2-stage ROW method up to order 3.

number	tree	order	
1	•	1	$b_1 + b_2 = 1$
2	/	2	$b_2\beta_2 = 1/2 - \gamma$
3	V	3	$b_2 a_2^2 = 1/3$
4	<	3	$0 = 1/6 - \gamma - \gamma^2$

Proposition 2.3

There exist ROW methods of order 2 with 2 stages and 2 function evaluations per step that satisfy in addition the conditions of all trees of order 3 except tree number 4. Where γ and a_2 are free parameters.

Proof

We obtain b_1 and b_2 from the linear system of equations

$$\begin{pmatrix} 1 & 1 \\ 0 & a_2^2 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/3 \end{pmatrix}$$

and from tree number (2)

$$\beta_2 = \frac{P_2(\gamma)}{b_2} .$$

The remaining coefficients can now be easily calculated

$$a_{21} = a_2$$

$$\beta_{21} = \beta_2$$

$$= a_{21} + \gamma_{21}$$

$$\gamma_{21} = \beta_{21} - a_{21}.$$

Chapter Three

Error Estimation For Single-Step Methods

In this chapter, we deal with the techniques used for the estimation of local truncation error for single-step methods. We also give a comparison, using ROW methods, of the two most commonly used techniques : the Richardson extrapolation technique and the Embedding technique.

3.10 Richardson Extrapolation

This has been the most popular method. The basic idea of the technique is that each basic step is done twice, once as two steps of size h to compute y_{n+1} and once as one step of size 2h to compute y^*_{n+1}, from y_{n-1}.

For a single-step method of order p. Henrici[25] shows that, assuming no previous errors have been made, the truncation error is given by

$$e_n = ch^{p+1} + O(h^{p+2}) \tag{3.1}$$

where c (assuming essentially constant) is called the error constant (corresponds to $N_1(1)$ in [39]). Then we have

$$\begin{aligned} y(x_{n+1}) - y_{n+1} &\doteq 2ch^{p+1} + O(h^{p+2}) = e_{n+1} \\ y(x_{n+1}) - y^*_{n+1} &= c(2h)^{p+1} + O(h^{p+2}), \end{aligned} \tag{3.2}$$

which implies

$$(3.3) \qquad e_{n+1} = \frac{y_{n+1} - y^{*}_{n+1}}{2^{p}-1}$$

Thus to apply Richardson extrapolation we compute two successive steps using steplength h, and then recompute over the double step using steplength 2h. The difference between the values for y so obtained, divided by $2^{p}-1$ is then an estimate of the local truncation error.

3.20 Embedding Methods

The Embedding method is a recent innovation due to England[16] and Fehlberg[18]. The basic idea is to perform the step from x_n to x_{n+1} with a p^{th} order and a $(p+1)^{th}$ order method to get an estimate of the error in the p^{th} order integration, where the p^{th} order method is obtained as a by-product of the $(p+1)^{th}$ order one. The coefficients γ, a_{ij}, b_i etc and therefore the values k'_i are the same for both methods. The different orders of the two methods lead to an estimate of the local truncation error of the p^{th} order method.

The algebra involved in obtaining such methods is formidable, but they have been obtained by England[16], Fehlberg[18] and Kaps-Rentrop [29] for ROW methods.

3.30 A Comparison of Richardson Extrapolation and Embedding Techniques for Stiff ODE's using ROW methods

In this section, we give a comparison of the two techniques for error estimates in the numerical solution of stiff ordinary differential equations. The method used to test these two techniques is the ROW method. The usual assumption that Richardson extrapolation is much worse than the embedding technique is not true as we will show. In (Kaps-Rentrop)[29], embedded pairs of ROW-methods are studied and algorithms are proposed based on a ROW method of order 4 which is used for step continuation and an embedded one of order 3 which is used for step-size control.

For explicit Runge-Kutta methods, preliminary calculations of Wanner[38], who compared Fehlberg's embedded order 4(5) formula RKF4 (using order 4 for step continuation and order 5 for error estimation) and a 4th order Runge-Kutta formula (using Richardson extrapolation), show that there is little difference between the two techniques. This contradicts the results published by Fehlberg[18].

For ROW methods, the situation seems to get worse for Richardson extrapolation because one has to consider the storage for the Jacobian and the solution of linear

systems. For one "double step" (i.e. $x_n \rightarrow x_n+h \rightarrow x_n+2h$), an embedded pair of ROW methods needs 2 LU decompositions, 2 Jacobian evaluations and up to 2v function evaluations. However, the usual Richardson extrapolation scheme needs 3 LU decompositions, 2 Jacobian evaluations and up to 3v function evaluations.

For the comparison, however, we derived a ROW method - called ROWR4 - of order 4 with 4 stages and 4 function evaluations having small truncation errors. This allows larger stepsizes. The Richardson extrapolation yields a fifth order error estimate instead of a third order estimate as in ROW4A[21]. Therefore, for the same number of LU-decompositions, ROWR4 requires slightly less total computing time than ROW4A. ROW4A has 50% more Jacobian calls than ROWR4, but fewer function evaluations, so that the total number of function evaluations is approximately the same for both methods, since the Jacobian was calculated with difference approximations.

3.31 Stability Properties

To study the stability properties of ROW-methods, the scalar test differential equation is used,

$$(3.4) \quad y' = \lambda y \qquad y(x_0) = y_0 = 1.$$

Since $f'(y)=\lambda$, one obtains for a ROW-method (2.6) the

following expression:

$$[1-\gamma h\lambda]k_i = h\lambda(y_0 + \sum_{j=1}^{i-1}(a_{ij}+\gamma_{ij})k_j) \quad i=1,\ldots,v.$$

Using the following abbreviations

$$z := \lambda h \qquad \beta_{ij} := a_{ij} + \gamma_{ij} \qquad u_i := y_0 + \sum_{j=1}^{i-1} a_{ij}k_j$$

we obtain:

$$k_i = \frac{z}{1-\gamma z}(y_0 + \sum_{j=1}^{i-1}\beta_{ij}k_j),$$

or, written in matrix notation

$$\underline{u} = y_0 + A\underline{k} \quad , \quad y_1 = y_0 + \underline{b}^T\underline{k}$$

$$\underline{k} = \frac{z}{1-\gamma z}(y_0 + B\underline{k})$$

where

$$\underline{k} = (k_1,\ldots,k_v)^T,$$

$$B = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \\ \beta_{21} & 0 & & & \\ \vdots & & & & \vdots \\ \beta_{v1} & \cdots & \beta_{v,v-1} & & 0 \end{pmatrix},$$

$$\underline{1} = (1,1,\ldots,1)^T,$$

$$\underline{u} = (u_1,\ldots,u_v)^T,$$

$$\underline{b} = (b_1,\ldots,b_v)^T,$$

Since $B^v=0$, we have

$$\underline{k} = \sum_{j=1}^{v} \left(\frac{z}{1-\gamma z}\right)^j B^{j-1} \mathbf{1}$$

and therefore

$$\underline{u} = \{ I + \sum_{j=1}^{v} \left(\frac{z}{1-\gamma z}\right)^j AB^{j-1} \} \mathbf{1} y_0 ,$$

$$y_1 = \{ 1 + \sum_{j=1}^{v} \left(\frac{z}{1-\gamma z}\right)^j b^T B^{j-1} \mathbf{1} \} y_0 ,$$

For the "special trees" with j nodes, the equations of condition read as follows :

$b^T B^{j-1} = P_j(\gamma)$, where

$$P_1(\gamma) = 1 ,$$

$$P_2(\gamma) = 1/2 - \gamma ,$$

$$P_3(\gamma) = \frac{1}{6} - \gamma + \gamma^2 ,$$

$$P_4(\gamma) = \frac{1}{24} - \frac{\gamma}{2} + \frac{3\gamma^2}{2} - \gamma^3 ,$$

. . .

as given in section 2.24. Thus, one obtains, for order $p \geq v$

$$y_1 = [1 + \sum_{j=1}^{v} \left(\frac{z}{1-\gamma z}\right)^j P_j(\gamma)] y_0 = R(z) y_0 .$$

$R(z)$ is called the stability function and depends on γ only. We note that $R(z)$ is a rational function:

$$R(z) = \frac{P(z)}{Q(z)} , \text{ where } P(z) = \sum_{i=0}^{\gamma} d_i z^i \text{ and}$$

$$Q(z) = (1-\gamma z)^v .$$

Proposition 3.1

Let $R(z) = \frac{\sum_{i=0}^{m} d_i z^i}{(1-\gamma z)^m}$ be an approximation to e^z of order $p \geq m$. Then :

$$(3.5) \quad d_k = (-\gamma)^k L_k^{(m-k)}(\tfrac{1}{\gamma})$$

where $L_k^{(\alpha)}(x)$ is the generalized Laguerre polynomial defined by(see Abramowitz[1]):

$$(3.6) \quad L_n^{(\alpha)}(x) = \sum_{j=0}^{n} (-1)^j \binom{n+\alpha}{n-j} \frac{x^j}{j!}, \quad \alpha > -1.$$

Proof : Since we can write

$$R(z) = e^z + O(z^{p+1}),$$

we have

$$(3.7) \quad P(z) = \sum_{i=0}^{m} d_i z^i = e^z (1-\gamma z)^m + O(z^{p+1}).$$

But

$$e^z = \sum_{j=0}^{\infty} \frac{z^j}{j!}$$

and $(1-\gamma z)^m = \sum_{k=0}^{m} \binom{m}{k} (-\gamma)^k z^k$

Therefore:

$$\begin{aligned} e^z (1-\gamma z)^m &= \sum_{l=0}^{\infty} z^l \sum_{j+k=l} \binom{m}{k} (-\gamma)^k \frac{1}{j!} \\ &= \sum_{l=0}^{\infty} z^l \sum_{j=0}^{l} \binom{m}{l-j} (-\gamma)^{l-j} \frac{1}{j!} \\ &= \sum_{l=0}^{\infty} z^l (-\gamma)^l \sum_{j=0}^{l} (-1)^j \binom{m}{l-j} (\tfrac{1}{\gamma})^j \frac{1}{j!} \end{aligned} \quad (3.8)$$

Using (3.7) and (3.8) and the definition of $L_n^{(\alpha)}(x)$ in (3.6) we obtain (3.5).

3.32 Error Estimate for The Test Equation (3.4)

Here, we compare the error estimates for ROW-methods using Richardson extrapolation and using embedding techniques for the test equation (3.4). Let $R_p(z)$ be the stability function of order p in proposition 3.1. Then, we define its error constant c, as in (3.1) :

$$(3.9) \quad e^z - R_p(z) = cz^{p+1} + O(z^{p+2}) .$$

Let y_1, y_2 be the numerical solutions at x_1, x_2 of (3.4) obtained by a ROW method of order p with v stages, $v \leq p$, and stepsize h, and y^*_1 the solution with stepsize 2h:

$$y_1 = R_p(z), \quad y_2 = R_p{}^2(z), \quad y^*_1 = R_p(2z) .$$

Let $E_p(z)$ be the error after two steps obtained by Richardson extrapolation. Then

$$E_p(z) = e^{2z} - (R_p(z))^2 = 2cz^{p+1} + O(z^{p+2}) .$$

Since $E_p(z)=(e^z+R_p(z))(e^z-R_p(z))$ and $e^z+R_p(z)=2+O(z)$. By eliminating c from the step using stepsize 2h, one gets

$$(3.10) \quad E_p(z) = e^{2z} - y_2 = \frac{y_2-y_1^*}{2^p-1} .$$

Thus $E_p(z)$ and $R_p(z)$ depend only on the coefficient γ.

For embedded method, the estimated error after one step with steplength h is given by

(3.11) $E(z) = R_4(z) - R_3(z)$.

3.33 On The Choice of γ For Rosenbrock Methods

In [39], Wanner has published some graphs containing a lot of information on the stability function. For $R_4(z)$, he obtains the graph containing

$\gamma_1 = .105663$ $T_1 = .105744$ $err_1 = .8912$

$\gamma_2 = .106055$ $T_2 = .217489$ $err_2 = .1494$

$\gamma_3 = .203843$ $T_3 = .553938$ $err_3 = .0762$

$\gamma_4 = .250000$

$\gamma_5 = .394339$

where it is A(0) stable between γ_1 and γ_2 and between γ_3 and γ_4. It is A-stable for γ greater than γ_5. T_i stands for the "Chebyshev" points, i.e. those γ-values where the maximal error

$$err = \max_{-\infty<x<0} |R(x)-e^x|$$

is locally minimized. From prop.(3.1)

$$\lim_{z\to\infty}|R_p(z)| = |L_s(\tfrac{1}{\gamma})| \quad \text{,where } L_s := L_s^{(0)}$$

If it is equal to zero, one has strong stability at infinity.

In (Kaps-Rentrop) [29], the graphs of the Laguerre polynomials $L_3(\frac{1}{\gamma})$ and $L_4(\frac{1}{\gamma})$ are plotted. The values of γ for which $L_4(\frac{1}{\gamma})=0$ are 0.106439, 0.220428, 0.572816 and an

uninteresting one at 3.100317. These values are close to the Chebyshev points T_i.

The graphs in Appendix (Appendix A) show for variable values of γ:

(a) For the Richardson extrapolation, we denote:

the stability function $R_4(z)$ ————————

the estimated error $E(z)$ — - — - — - — -

the actual error $e^{2z} - (R_4(z))^2$ —— + —— + —— + ——

(b) For Embedded methods, we include the following curves:

the stability function $R_4(z)$ ————————

and $R_3(z)$ - - - - - - - - - -

the actual errors $e^z - R_4(z)$ —— + —— + —— + ——

and $e^z - R_3(z)$ .....

the estimated error $E(z)$ — - — - — - — - .

The graph of $R_4(z)$ for $\gamma=0.106439$ with $L_4(\frac{1}{\gamma})=0$ shows that $|R_4(-x)|>1$ for $-70<x<-25$. A method with that values of γ is not A(0)-stable. Since $R_3(z)$ is unstable at $\gamma \approx 0.1$ (see Kaps-Rentrop[29]), we omit the corresponding points for embedded methods. It is surprising that for the values of γ with a small error constant (see Wanner[39]), these are the smallest values of γ within each region of A(0)-stability) we may have larger global errors. This behaviour possibly explains why it is good to use L(α)-stable methods for very stiff problems, since the Chebyshev points are close to zeros of $L_4(\frac{1}{\gamma})$ and for these

problems, the accuracy for small z is not so important.

In next section, we will derive and compare ROW methods of different γ values.

3.34 ROWR4 - ROW Methods Of Order 4 Using Richardson Extrapolation

In this section, ROWR4, a ROW method of order 4 is derived which uses Richardson extrapolation for error estimates, and then the method is compared with the A-stable ROW methods called ROW4A (see Gottwald-Wanner[21]) which uses the embedding technique for error estimate.

As we have more degrees of freedom for a Richardson type ROW method, we try to make the error terms of order 5 as small as possible. Therefore, we derive in the following a ROW method of order 4 using 4 stages and 4 function evaluations. (There exists, however, a ROW method of order 4 using 3 stages and 3 function evaluations, due to Hairer. This method is called MORO4 in Kaps [28], [29]).

In the following table, we list the order conditions for a 4-stage ROW method up to order 5 , which were derived in Chapter two.

number	tree	order	
1		1	$b_1+ b_2+ b_3 + b_4 = P_1$
2		2	$b_2\beta_2+b_3\beta_3+b_4\beta_4 = P_2$
3		3	$b_2a_2^2+b_3a_3^2+b_4a_4^2 = P_3$
4		3	$b_3\beta_{32}\beta_2+b_4\beta_{42}\beta_2+b_4\beta_{43}\beta_3 = P_4$
5		4	$b_2a_2^3+b_3a_3^3+b_4a_4^3 = P_5$
6		4	$b_3a_3a_{32}\beta_2+b_4a_4a_{42}\beta_2+b_4a_4a_{43}\beta_3 = P_6$
7		4	$b_3\beta_{32}a_2^2+b_4\beta_{42}a_2^2+b_4\beta_{43}a_3^2 = P_7$
8		4	$b_4\beta_{43}\beta_{32}\beta_2 = P_8$
9		5	$b_2a_2^4+b_3a_3^4+b_4a_4^4 = P_9$
10		5	$b_3a_3^2a_{32}\beta_2+b_4a_4^2a_{42}\beta_2+b_4a_4^2a_{43}\beta_3 = P_{10}$
11		5	$b_3(a_{32}\beta_2)^2+b_4(a_{42}\beta_2)^2+b_4(a_{43}\beta_3)^2 = P_{11}$
12		5	$b_3a_4a_{32}a_2^2+b_4a_4a_{42}a_2^2+b_4a_4a_{43}a_3^2 = P_{12}$
13		5	$b_4a_{43}\beta_{32}\beta_2 = P_{13}$
14		5	$b_3\beta_{32}a_2^3+b_4\beta_{42}a_2^3+b_4\beta_{43}a_3^3 = P_{14}$
15		5	$b_4\beta_{43}a_3a_{32}\beta_2 = P_{15}$
16		5	$b_4\beta_{43}\beta_{32}a_2^2 = P_{16}$
17		5	$0 = P_{17}$

Proposition 3.2

There exist ROW methods of order 4 with 4 stages and 4 function evaluations that satisfy in addition the conditions for all trees of order 5, except 12 and 17. The coefficients γ, a_3 and a_4 are, in general, free parameters.

Proof

We replace the equations 6, 10, 11, 15 by the simplifying assumptions (see section 2.24)

SA2 $\quad a_2 = 2\gamma$,

SA3 $\quad a_{32}\beta_2 = a_3(a_3/2-\gamma)$,

SA4 $\quad a_{42}\beta_2 + a_{43}\beta_3 = a_4(a_4/2-\gamma)$.

If the other equations 1-9 and 14 are satisfied, SA2-SA3 are equivalent to 6, 10, 11, 15. From equations 1, 3, 5, 9, we obtain b_i from the linear system of equations:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & a_2^2 & a_3^2 & a_4^2 \\ 0 & a_2^3 & a_3^3 & a_4^3 \\ 0 & a_2^4 & a_3^4 & a_4^4 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/3 \\ 1/4 \\ 1/5 \end{pmatrix}.$$

From the linear system (equations 7 and 14)

$$\begin{pmatrix} a_2^2 & a_3^2 \\ a_2^2 & a_3^3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} P_7 \\ P_{14} \end{pmatrix},$$

where $x_1 := b_3\beta_{32} + b_4\beta_{42}$ and $x_2 := b_4\beta_{43}$ one gets :

$$\beta_{43} = \frac{x_2}{b_4}$$

Equation 16 yields

$$\beta_{32} = \frac{P_{16}}{x_2 a_2^2}$$

From equation 13 it follows

$$a_{43} = \frac{P_{13}}{b_4 a_4 \beta_{32} \beta_2}.$$

and from the simplifying assumptions

$$a_{32} = a_3(a_3/2-\gamma)/\beta_2,$$

$$a_{42} = [a_4(a_4/2-\gamma)-a_{43}\beta_3]/\beta_2.$$

The remaining coefficients can now be easily calculated:

$$a_{21} = a_2, \quad a_{31} = a_3 - a_{32}, \quad a_{41} = a_4 - a_{42} - a_{43},$$
$$\beta_{21} = \beta_2, \quad \beta_{31} = \beta_3 - \beta_{32}, \quad \beta_{41} = \beta_4 - \beta_{42} - \beta_{43},$$
$$\gamma_{ij} = \beta_{ij} - a_{ij} \quad (i>j).$$

With Proposition 3.2 we calculated a ROWR4 as follows : We choose γ. Then, the parameters a_3 and a_4 were varied between $0<a_i<1$ in order to obtain the following conditions:

1) small values for non-vanishing error terms of order 5,
2) positive coefficients m_j (see below)
3) coefficients a'_{ij} and c_{ij} with small absolute value

In order to adapt ROW method (2.7) for numerical computation, we transform the formula (2.7), by putting

$$c_{ij} := \sum_{k=1}^{i-1} \gamma_{ik}(\delta_{kj}-c_{kj})/\gamma,$$

$$a'_{ij} := \sum_{k=1}^{i-1} a_{ik}(\delta_{kj}-c_{kj}) \quad (i,j=1,\ldots,v),$$

$$m_j := \sum_{k=1}^{v} b_k(\delta_{kj}-c_{kj})$$

,into an equivalent formula

$$E = I - \gamma h f'(y_0),$$

$$(3.12) \qquad Ek_i = f(y_0 + h\sum_{j=1}^{i-1} a'_{ijkj}) + \sum_{j=1}^{i-1} c_{ijkj} \quad (i=1,\ldots,v),$$

$$y_1 = y_0 + h\sum_{i=1}^{v} m_i k_i$$

As we have seen in the last section, roughly speaking, smaller values of γ give better error constants but increase instability, thus there is no optimal choice which optimizes simultaneously error constant and stability. A reasonable choice of the parameters is γ=0.231, a_3=0.83 and a_4=0.40 for ROWR4. Thus we have:

γ= 0.231

$$(3.13) \qquad \begin{aligned} a'_{21} &= .462000000000000 & c_{21} &= -.602873560260317 \\ a'_{31} &= .647090786918081 & c_{31} &= 3.40797125852141 \\ a'_{32} &= .510943600770895 & c_{32} &= -.569789627235356 \\ a'_{41} &= .796361654523372 & c_{41} &= -4.09977762735509 \\ a'_{43} &= -.158807672550906 & c_{43} &= .433701554349537 \\ m_1 &= 1.46733176350210 \\ m_2 &= 2.09384419897536 \\ m_3 &= 0.0836917542591757 \\ m_4 &= 0:649641577060932. \end{aligned}$$

The detail results of the comparison of ROWR4 and ROW4A are given in Chapter 5 and (Poon-Kaps-Bui)[34].

3.35 ROWR4 Methods With Different Parameters

ROW4A is implemented as BACk-STEP algorithm (see [21]

Gottwald-Wanner). i.e. for each single-step of integration, the algorithm is designed as

i) after a failure of the error test, not only the current step, but also one previous step is repeated with smaller step-size;

ii) the final result for t=tend is only accepted when at least one more step beyond tend has been tested.

It is because, for Rosenbrock-type method the computation involves the calculation of the Jacobian of f(y) at the (local) initial point only and therefore approximate the error of the previous step rather than the current one. This can have bad consequences in regions where the solutions change from a smooth behaviour to rapid movements, so that one accepted step with large local error falsifies the whole computation. An example is problem D5 of Enright et al [17], where the error function increases rapidly at the end of the integration interval and many programs give incorrect results. (see Fig 4.1). Thus back-step algorithm increases the reliability of a program.

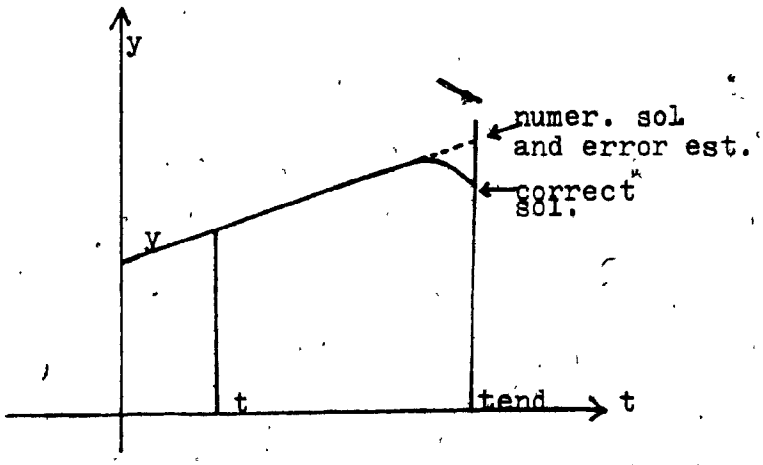


Fig 4.1

In ROWR4, however, each y_{n+1} is obtained by the integration of two successive steps from y_{n-1}. After a failure of the error test, we repeat the integration from y_{n-1} with smaller stepsize (i.e. repeat two steps). Stetter[36] suggests that by putting one of the free parameter a_i equal to one in ROWR4, for a one step integration from y_0 to t=tend, we actually evaluate the last point at t=tend. This is similar to the idea of back-step algorithm. Thus we have the following ROWR4 methods.

$$\gamma = 0.231 \quad a_3 = 1.00 \quad a_4 = 0.68$$

$$a'_{21} = 0.462000000000000$$

$$a'_{31} = 0.668995886482497$$

$$a'_{32} = 0.833498050984662$$

$$a'_{41} = 0.975086972773952$$

$$a'_{42} = 0.976581368838382$$

$$a'_{43} = -0.0886972448493171$$

(3.14)

$$c_{21} = -0.602873560260318$$

$$c_{31} = 6.48993262908732$$

$$c_{32} = 0.527391990745225$$

$$c_{41} = -3.72425321659562$$

$$c_{42} = -3.90639493869092$$

$$c_{43} = 0.143013080533142$$

$$m_1 = 1.02929062691478$$

$$m_2 = 1.60010560784412$$

$$m_3 = 0.06065452676420888$$

$$m_4 = 0.356511499634932$$

In order to study the behaviour of ROWR4 on Υ, we obtain the following ROWR4 methods with γ=0.395(A-stable), γ=0.572816(L-stable, see Bui[6]), γ=0.217489(Chebyshev point), γ=0.553938(Chebyshev point). The detail results of the comparison of such methods are given in Chapter 5.

$\gamma = 0.395 \quad a_3 = 0.97 \quad a_4 = 0.37$

$a'_{21} = 0.790000000000000$

$a'_{31} = 0.772970049618904$

$a'_{32} = 0.0239827078467521$

$a'_{41} = 0.196003759952297$

$a'_{42} = -0.124790834350893$

$a'_{43} = 0.304932969741333$

(3.15) $c_{21} = 7.21550058651026$

$c_{31} = 4.94939557647087$

$c_{32} = -0.245472256241477$

$c_{41} = -6.41238267605767$

$c_{42} = 0.353500784152382$

$c_{43} = -1.12505900920242$

$m_1 = 0.310373472349139$

$m_2 = 0.185504231671458$

$m_3 = 0.576455119110363$

$m_4 = 0.447358555466664$

$\gamma = 0.553938 \quad a_3 = 0.29 \quad a_4 = 0.61$

$a'_{21} = 1.10787600000000$

$a'_{31} = 0.457308621758682$

$a'_{32} = -0.0467803557081104$

$a'_{41} = 0.803515205428698$

$a'_{42} = -0.0554562448022975$

$a'_{43} = 0.04132241961552219$

(3.16) $c_{21} = 2.57647177380644$

$c_{31} = -0.779818723249710$

$c_{32} = -0.0289328010304023$

$c_{41} = -1.28580104007137$

$c_{42} = -0.104101897863716$

$c_{43} = -0.946929696920999$

$m_1 = 1.00324582150009$

$m_2 = 0.136849203813239$

$m_3 = 0.485989196455463$

$m_4 = 0.714779193084515$

$\gamma = 0.572816$ $a_3 = 0.14$ $a_4 = 0.52$

$a'_{21} = 1.14563200000000$

$a'_{31} = 0.235327766131218$

$a'_{32} = -0.0275637732112589$

$a'_{41} = 0.724883222781744$

$a'_{42} = -0.0637558470789723$

$a'_{43} = 0.0232376182044381$

(3.17) $c_{21} = 2.45844400186398$

$c_{31} = -0.402370457549341$

$c_{32} = 0.0214682448956575$

$c_{41} = 0.174769430483443$

$c_{42} = 0.00758440015727501$

$c_{43} = -2.56058902537767$

$m_1 = 0.932932535704117$

$m_2 = 0.114867738718394$

$m_3 = 0.325620288181542$

$m_4 = 1.05693766084662$.

$\gamma = 0.217489 \quad a_3 = 0.80 \quad a_4 = 0.33$

$a'_{21} = 0.434978000000000$

$a'_{31} = 0.587637998919792$

$a'_{32} = 0.458976779271902$

$a'_{41} = 0.522868278105480$

$a'_{42} = 0.334207131997375$

$a'_{43} = -0.0718050398433548$

(3.18) $c_{21} = -0.537314281090455$

$c_{31} = 3.71324061307182$

$c_{32} = 0.272903404031621$

$c_{41} = -2.74313959118121$

$c_{42} = -1.76756488805389$

$c_{43} = 0.197614036356602$

$m_1 = 1.27316376962252$

$m_2 = 1.48560183712702$

$m_3 = 0.248886754860845$

$m_4 = 1.34924870375566$

Chapter Four

A Modified Richardson Extrapolation Scheme For The Error Estimate In Rosenbrock Methods

The Richardson extrapolation scheme described in Chapter Three has been widely used for the estimation of local truncation errors in single-step methods for the numerical solutions of systems of ODE's. This procedure is fairly accurate for step-control purposes, but it involves a considerable increase in computational effort. In this chapter, we describe a simple modified Richardson extrapolation scheme for estimating the local truncation error in a Rosenbrock procedure for the numerical solution of stiff differential equations. This approach was first described by Cash[10] for a third order A-stable process. However, Cash's paper contains errors[3]. The aim of this chapter is to present the correct version of an efficient third order A-stable method.

4.10 Rosenbrock Procedures With Built-In Error Estimates

Several algorithms of the general Rosenbrock procedure of the form (2.4) have been proposed for the numerical integration of stiff systems of equation (1.1) but, apart from the algorithm due to Haines[22], attemps to obtain an error estimate applicable to systems of equations which may be used in a step control procedure are noticeable by their absence. Moreover, the error estimate used by Haines is

strictly valid only for linear systems, which may perform badly and produce diastrous results on certain classes of problems[11] (Cash-Liem).

Putting $v=3$ and $c_{ij}=0$ in equation (2.6), we obtained the order conditions for the method :

$$b_1 + b_2 + b_3 = 1$$

$$b_1 a_{21} + b_3(a_{31} + a_{32}) = \frac{1}{2} - \gamma \tag{4.1}$$

$$b_2 a_{21}^2 + b_3(a_{31} + a_{32})^2 = \frac{1}{3}$$

$$b_3 a_{21} a_{32} = \gamma^2 - \gamma + \frac{1}{6}$$

Let us denote this third order process by

$$R_3 = (b_i, \gamma, a_{ij}, h),$$

then

$$y(x_{n+1}) = R_3 + 2h^4 A(ff^{(3)} + \frac{B}{A} f'f'' + \frac{C}{A} f'f^{(3)}) \tag{4.2}$$

where

$$A = \gamma^3 - \frac{3}{2}\gamma^2 + \frac{1}{2}\gamma - \frac{1}{24}$$

$$B = -\frac{1}{2}\gamma + \frac{1}{6} - b_3 a_{32} a_{21}(\frac{1}{2}a_{21} + a_{31} + a_{32})$$

$$C = \frac{1}{24} - \frac{1}{6}a_{21}^3 b_3 - \frac{1}{6}(a_{31} + a_{32})^3 b_3$$

The second term in equation (4.2) is the principal term of the local truncation error.

The Richardson extrapolation scheme requires to carry out twice integration procedure (b_i, γ, a_{ij}, h) and once procedure $(b_i, \gamma, a_{ij}, 2h)$. Each k_i must be evaluated twice : for steplength h and for step-length 2h. We note that when Rosenbrock procedures are employed to solve stiff systems of differential equations, the solution of the linear systems of algebraic equations, for obtaining the k_i, forms a large proportion of the total computation cost. Thus Richardson extrapolation scheme would require about 50% extra computation in order to obtain the solution with step-length 2h. However, if we introduce three new parameters $\bar{b}_1$, $\bar{b}_2$, $\bar{b}_3$, so that $\bar{R}_3$ which is defined by $(\bar{b}_i, \gamma/2, a_{ij}/2, 2h)$ is of the same order of accuracy of R_3, then k_i are the same for both integration procedures and a negligible additional amount of computation is required to calculate the solution with step-length 2h. For the formula R_3 to be of third order, the coefficients must satisfy :

$$\begin{aligned} &\bar{b}_1 + \bar{b}_2 + \bar{b}_3 = 1 \\ &\bar{b}_2 a_{21} + \bar{b}_3 (a_{31} + a_{32}) = 1 - \gamma \\ &\tfrac{1}{4}\bar{b}_2 a_{21}^2 + \tfrac{1}{4}\bar{b}_3 (a_{31} + a_{32})^2 = \tfrac{1}{3} \\ &\tfrac{1}{4}\bar{b}_3 a_{21} a_{32} = \tfrac{1}{4}\gamma^2 - \tfrac{1}{2}\gamma + \tfrac{1}{6} \end{aligned} \tag{4.3}$$

Also, we have

$$y(x_{n+1}) = \bar{R}_3 + 16h^4\bar{A}\left(ff^{(3)} + \frac{\bar{B}}{\bar{A}} f' f'' + \frac{\bar{C}}{\bar{A}} f'' f^{(3)}\right) \tag{4.4}$$

where $\bar{A}$, $\bar{B}$, $\bar{C}$, are obtained from A, B, C respectively by substituting γ and a_{ij} by $\gamma/2$ and $a_{ij}/2$. Equations (4.1) and (4.3) now contain 10 unknowns in 8 equations, we will choose these two free parameters so that expressions (4.2) and (4.4) may be used to provide an estimate of the local truncation error. To this aim, we impose the condition that:

$$(4.5) \quad \frac{B}{A} = \frac{\bar{B}}{\bar{A}} = \alpha \quad \text{and} \quad \frac{C}{A} = \frac{\bar{C}}{\bar{A}} = \tau$$

so that equations (4.2) and (4.4) become

$$y(x_{n+1}) = R_3 + 2h^4 A\beta$$

$$= \bar{R}_3 + 16h^4 \bar{A}\beta$$

where

$$\beta = ff'^{(3)} + \alpha f' f'' + \tau f^2 f^{(3)}$$

Therefore:

$$R_3 - \bar{R}_3 = 24h^4 (8\bar{A} - A)$$

The local truncation error is:

$$e_n = 2h^4 A\beta$$

$$(4.6) \quad = \frac{\mu}{1-\mu}(R_3 - \bar{R}_3)$$

where

$$(4.7) \quad \mu = \frac{A}{8\bar{A}} = \frac{\gamma^3 - \frac{3}{2}\gamma^2 + \frac{1}{2}\gamma - \frac{1}{24}}{\gamma^3 - 3\gamma^2 + 2\gamma - \frac{1}{3}}$$

Equation (4.1) (4.3) and (4.5) give 10 unknowns in 10 equations. By solving this set of non-linear algebraic equations, the results shown on formula (4.8) define a new third-order A-stable scheme with modified Richardson extrapolation for error estimate - called ROB3A.

$$
\begin{aligned}
&\gamma = 0.9712451501 \\
&b_1 = -2.7478803512 \quad b_1 = 9.4141605053 \\
&b_2 = 0.08170187961 \quad b_2 = 0.3723461442 \\
&b_3 = 3.6661784716 \quad b_3 = -8.7865066495 \\
&a_{21} = -1.9372312845 \\
&a_{31} = -0.06583223722 \\
&a_{32} = -0.019534450510
\end{aligned}
\tag{4.8}
$$

The algorithms are illustrated by direct application to test equations in Chapter Five. The detail results of the comparison of ROB3A with the original Cash's algorithm and other methods of the same class are given in [7](Bui-Poon).

Chapter Five

Comparing Numerical Methods For Stiff Systems of ODE's

A comparison of methods for solving non-stiff systems of ordinary differential equations was described in [27] (Hull et al). A comparison of some traditional methods : GEAR (Gear's variable-order method based on backward differentiation multistep formulas[20]), SDBASIC (variable-order method based on second derivative multistep formula), GENRK (a fourth-order generalized Runge-Kutta method developed by Lawson and Ehle[32]), TRAPEX (fourth order method based on the trapezoidal rule with extrapolation), IMPRK (fourth-order, two stage, A-stable implicit Runge-Kutta method), for solving stiff systems of ODE's was given in [17] (Enright-Hull-Lindberg). In this chapter we compare the methods proposed in Chapter Three and Chapter Four. The basis of a fair comparison is based on the technique proposed by Enright et al [17]. We begin in the next section with a discussion of some important conceptual aspects of testing, followed by the section which we shall discuss the comparison criteria. The following section then will give a brief description of the methods for testing. We summarize results obtained with different methods in the last section.

5.10 Conceptual Aspects

In [17](Enright et al), the comparison of different methods for solving stiff ODEs were made fully automatic by designing a program DETEST to carry out the tests. One of the major practical difficulties in making a numerical comparison of integration methods is that of choosing a truly representative class of test problems. This difficulty has, however, now been largely removed for small systems at least as a result of DETEST. There are twenty-five excellently selected differential equation systems (classified as CLASS A to CLASS E). In addition to these 25 problems, we also include the five problems - called CLASS F, of somewhat harder complexity, described by Gottwald-Wanner[21], in our tests. The CLASS F test problems are shown in Appendix (Appendix-B).

It is obvious that for any algorithm reliability is of major importance. A program is reliable if it obtains the solution to the requested accuracy, whenever this is possible. Equally important is a reasonable degree of efficiency. Here, we will restrict our attention primarily to measures of cost and reliability of different methods. In [27], measures of cost in terms of total time and number of function calls were proposed. However, for numerical methods for stiff systems, the number of Jacobian evaluations and thus the number of matrix inversion calls

are also very important factors in measuring the costs. the reliability statistics are measured in terms of the maximal gobal truncation error for solving a problem.

For the control of step-size, we adopt the strategy described in [19](Gear) for Richardson extrapolation as follows : The errors in each component at each step are computed and divided by the numbers YMAX(I). These are set initially to one, and are updated to contain the maximum value of Y(I) that has been just computed. The value of h for the next step is computed to make h as large as necessary to make the next scaled error equal to TOL.

```
YS = MAX(Y(I),YMAX(I))

ERRMAX = MAX (ERROR(I)/(TOL*YS)
         , I=1,v

HNEW = 2.00*H      for ERRMAX=0

HNEW = H*ERRMAX**(-0.25)*0.99      for ERRMAX>0
```

The factor 0.99 serves to keep h slightly smaller so that if the truncation error on the next step is slightly larger, the h recommended will be small enough. If the scale error is greater than TOL, the step is rejected and repeated with the newly recommended h. The algorithm keeps the relative error approximately constant in the max-norm.

5.20 Methods

We have basically included four methods in our tests. ROW4A : the fourth order A-stable ROW method using embedding technique for error estimate (Gottwald-Wanner[21]), ROWR4 : the fourth order A-stable ROW method using Richardson extrapolation for error estimate (formula (3.13)), ROB3A : third order A-stable Rosenbrock methods with modified Richardson extrapolation technique for error estimate (formula (4.9)), GEAR : the January 13, 1975 version of Hindmarsh's code for Gear's BDF-formulas[26]. Besides, we also compare the ROWR4 of different parameters : $\gamma = 0.231$ (formula (3.14)), $\gamma = 0.395$ (formula (3.15)), $\gamma = 0.553938$ (formula (3.16)), $\gamma = 0.572816$ (formula (3.17)), and $\gamma = 0.217489$ (formula (3.18)).

All the methods were coded in FORTRAN in single precision. Copies of these codes can be obtained from the author. Each method requires a matrix inversion routine. Of course, for efficiency in a production code, this should be replaced with the equivalent L-U decomposition followed by back substitution where necessary. Thus we choose the subroutines DEC and SOL designed by Moler[20] for solving the linear systems in all methods.

The Jacobian matrix at each step was calculated with difference approximation.

Each integration subroutine has the following eleven paremeters :

(N,FCN,T,Y,TEND,TOL,HMAX,TIMEMX,H,IPAS,IREP)

The input parameters are

N .... Dimension of the system;

FCN ... External name of subroutine FCN(T,Y,F) to compute the functions of the differential equation y'=f(y).

T ..... Initial value for T;

Y ..... Vector of dimension at least N containing initial values of Y;

TEND .. Value of T for which solution is desired;

TOL ... Relative error tolerance per step;

HMAX .. Maximal stepsize;

TIMEMX Maximal computing time allowed before an error return.

H ..... Initial guess for step size; the program adjusts stepsize then automatically; IF(H.EQ.0), H is taken as 1.e-2.

The output parameters are

T ..... T=TEND on normal return, otherwise point up to which solution has been computed;

Y ..... Computed solution vector;

TIMEMX Used computing time on successful return; otherwise something negative;

H ..... Successful size of step size in last steps;

IPAS .. Number of computed steps;

IREP .. Number of repeated steps.

5.30 Comparison Criteria

The comparison criteria were chosen to reflect both the cost (or efficiency) and the reliability of a method. We distinguish five components of the total cost : the total computing time to solve a problem, number of steps, number of function evaluations, number of Jacobian evaluations, and the number of matrix inversions. The measure of reliability of a method is based on the maximal global truncation errors over the whole integration interval and all components. Each method was tested on the 30 test equations at three different tolerance (TOL) 10^{-2}, 10^{-3}, 10^{-4}. The calculations were performed at the CDC Cyber 174 at Concordia University having approximately 14 digits accuracy (48 bits mantissa), at a cost of about $500/hr.

5.40 Results And Discussion

TOL - tolerance

TIME - total computing time

FCN CALLS - number of function calls

JAC CALLS - number of Jacobian evaluations

INV CALLS - number of matrix inversion calls

TF - total number of function evaluations (includes Jacobian with finite difference)

NO OF STEPS - number of total accepted steps (one double step in ROWR4 is counted as two steps)

Summaries over all problems*

METHOD	TOL	TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS	MAXIMUM ERROR
GEAR	10**-2	10.642	-	498	498	7038	3724	.245E+05
	10**-3	13.552	-	592	292	8495	4540	.923E+05
	10**-4	18.059	-	669	669	10612	5615	.213E+06
	OVERALL SUMMARY	42.253	-	1759	1759	26145	13879	.330E+06
ROW4A	10**-2	7.761	4797	1587	1599	10782	1367	.226E+03
	10**-3	10.402	6201	2037	2067	14045	1827	.379E+03
	10**-4	16.086	9555	3144	3185	21766	2948	.179E+04
	OVERALL SUMMARY	34.249	20553	6768	6851	46593	6142	.240E+04
ROWR4 (3.13)	10**-2	9.087	7764	1294	1975	12025	808	.159E+07
	10**-3	9.228	7584	1264	1896	12110	874	.167E+04
	10**-4	13.982	11316	1886	2829	18125	1298	.980E+03
	OVERALL SUMMARY	32.297	26664	4444	6700	42260	2980	.159E+07
ROWR4 (3.14)	10**-2	7.703	6552	1092	1638	10228	712	.489E+05
	10**-3	9.110	7488	1248	1072	11978	864	.138E+04
	10**-4	14.194	11496	1916	2874	18384	1298	.853E+03
	OVERALL SUMMARY	31.007	25536	4255	5584	40590	2874	.511E+05
ROWR4 (3.15)	10**-2	7.995	7092	1182	1773	10929	726	.490E+07
	10**-3	10.227	8592	1432	2148	13604	958	.101E+04
	10**-4	14.621	12336	2056	3084	19416	1412	.180E+04
	OVERALL SUMMARY	32.843	28020	4670	7005	43949	3096	.490E+07

ROWR4 (3.16)	10**-2	7.893	6612	1102	1653	10393	700	.198E+07
	10**-3	11.607	9612	1602	2403	15261	1076	.142E+04
	10**-4	18.031	14688	2448	3672	23470	1638	.150E+04
	OVERALL SUMMARY	37.531	30912	5152	7728	49124	3414	.198E+07
ROWR4 (3.17)	10**-2	7.817	6672	1112	1668	10436	704	.327E+07
	10**-3	11.580	9852	1642	2463	15525	1090	.135E+04
	10**-4	18.526	15018	2518	3777	24125	1688	.148E+04
	OVERALL SUMMARY	37.923	31632	5272	7980	60086	3482	.327E+07
ROWR4 (3.18)	10**-2	7.478	6360	1060	1590	9942	674	.148E+04
	10**-3	8.926	7392	1232	1848	11800	854	.101E+04
	10**-4	13.214	10740	1790	2685	17155	1244	.553E+03
	OVERALL SUMMARY	29.618	24492	4082	6123	38897	3772	.304E+04
ROB3A	10**-2	7.272	5280	1760	1760	12164	1106	.175E+03
	10**-3	13.215	9552	3184	3184	21906	2034	.467E+03
	10**-4	24.016	17670	5890	5890	40150	4040	.760E+03
	OVERALL SUMMARY	44.503	32502	10834	10834	74220	7180	.139E+04

*The detail results of the testing are shown in Appendix (Appendix C).

One can immediately see from the results of the comparison of ROWR4 and ROW4A that there is little difference between the two error estimation techniques. This contradicts the usual assumption that Richardson extrapolation is much worse than embedded method. It seems to be more efficient to derive schemes requiring as few function or Jacobian evaluations as possible and to use them with an Richardson extrapolation error estimate rather than to derive methods using embedded method as error estimates. Although ROW methods are always slightly slower than GEAR, from the plots of the maximal global errors, they give more reliable results than GEAR for all 30 test equations. ROB3A is also comparable with GEAR, though it is only a third-order method. From the comparison of ROW methods with different coefficients, the method with $\gamma=0.395$ seems to be the best despite of the bad behaviour of the stability function. It is surprising that the L-stable method($\gamma=0.572816$) does not show superior to other methods. Very recently, Shampine[43] made a critical evaluation of the set of test problems. He claims that most of the problems are not stiff enough for testing. This may explain the results we obtained. However, it is still not clear at this time that a L-stable method is better than other methods. Recently, we have developed a new L-stable ROWR4 method. We plan to test it with some more stiff problems. Hopefully, we shall find out the importance of L-stability.

6. SUGGESTIONS FOR FUTURE WORK:

The following suggestions could well form the basis for further work in connection with the numerical integration of stiff systems of ordinary differential equations:

1. Attempt to develop and obtain the plots of the stability function $R(z)$ of ROW methods for the nonlinear case, in order to study the stability behaviour of ROW methods with different coefficients for nonlinear ODE's.

2. Attempt to reduce both the number of LU-decomposition and Jacobian evaluations in Runge-Kutta methods using a time-lagged Jacobian or an arbitrary matrix instead of the exact Jacobian, or use the updating technique which is usually used in determination of the eigenvalues of a real general matrix.

3. Attempt to develop a new modified Richardson extrapolation scheme which uses the two k_i coefficients calculated in two successive steps with steplength h(instead of one of them as described in Chapter Four) to estimate the k_i for steplength 2h.

4. Attempt to compare all newly proposed techniques for

stiff systems. Many new techniques have been proposed recently, but inclusion of actual implementations of such techniques in a software package should be preceded by a careful evaluation of their expected performance. Such a comparative study has not yet been undertaken.

7. REFERENCES

1. M. Abramowtiz and I. A. Stegun, Handbook of Mathematical Functions, New York, Dover Publ. Inc., (1970).

2. O. Axelsson, A Class of A-Stable Methods, BIT, 9, 185-199 (1969).

3. G. Bjurel, G. Dahlquist, B. Lindberg, S. Linde and L. Oden, Survey of Stiff Ordinary Differential Equations, Royal Institute of Technology, Computer Science Report NA 70.11 (1970).

4. T. D. Bui, Errata and Comments on : "Semi-Implicit Runge-Kutta Procedures with Error Estimates for the Numerical Integration of Stiff Systems of Ordinary Differential Equations", J. ACM, 24, 623 (1977).

5. T. D. Bui and T. R. Bui, Numerical Methods for Extremely Stiff Systems of Ordinary Differential Equations, Appl. Math. Modellling 3, 355-358 (1979).

6. T. D. Bui, Some A-Stable and L-Stable Methods for the Numerical Integration of Stiff Ordinary Differential Equations, J. Assoc. Comput. Math. Vol. 26, No.3, 483-493 (1979).

7. T. D. Bui and S. Poon, On the Computational Aspects of Rosenbrock Procedures with Built-in Error Estimates for Stiff Systems, BIT, 21, 78-89, (1981).

8. J. C. Butcher, Coefficients for the Study of Runge-Kutta Integration Processes, J. Austral. Math. Soc. 3, 185-201 (1963).

9. J. C. Butcher, Implicit Runge-Kutta Processes, Math. Comp. 18, 50-64 (1964).

10. J. R. Cash, Semi-Implicit Runge-Kutta Procedures with Error Estimates for the Numerical Integration of Stiff Systems of Ordinary Differential Equations, J. ACM, 23, 455-460 (1976).

11. J. R. Cash and C. B. Liem, On the Computational Aspects of Semi-Implicit Runge-Kutta Methods, Computer J. Vol. 21, No.4 383-385 (1977).

12. F. H. Chipman, Numerical Solution of Initial Values Problems Using A-Stable Runge-Kutta Processes, Thesis, Dept. of AACS, University of Waterloo (1971).

13. C. F. Curtiss and J. O. Hirschfelder, Integration of

Stiff Equations, Proc. Nat. Acad. Sci. U.S.A., 38, 235-243 (1952).

14. G. Dahlquist, A Special Stability Problem for Linear Multistep Methods, BIT 3, 27-43 (1963).

15. B. L. Ehle, On Pade Approximations to the Exponential Function and A-Stable Methods For the Numerical Solution of Initial Value Problems, Res. Rept. CSRR 2010. University of Waterloo (1969).

16. R. England, Error Estimates for Runge-Kutta Type Solution to Systems of O.D.E's. Comput. J. 12, 166-169 (1969).

17. W. H. Enright, T. E. Hull, B. Lindberg, Comparing Numerical Methods for Stiff Ordinary Differential Equations, BIT 15, 10-48 (1975).

18. E. Fehlberg, Classical Fifth, Sixth, Seventh and Eight Order Runge-Kutta Formulas with Stepsize Control, NASA T.R. 287 (1968).

19. C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice Hall, Englewood Cliffs, New Jersey (1971).

20. C. W. Gear, Algorithm 407, DIFSUB for Solution of Ordinary Differential Equations, CACM 14, 185-190 (1971).

21. B. A. Gottwald, G. Wanner, A reliable Rosenbrock Integrator for Stiff Differential Equations, Submitted to Computing, (1980).

22. C. F. Haines, Implicit Integration Processes with Error Estimates for the Numerical Solution of Differential Equations, Computer J., 12, 183-187 (1968).

23. E. Hairer and G. Wanner, Multistep - Multistage - Multiderivative Methods for Ordinary Differential Equations, Computing 11, 287-303 (1973).

24. E. Hairer and G. Wanner, On the Butcher Group And General Multi-value Methods, Computing 13, 1-15 (1974).

25. P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, J. Wiley and Sons, New York (1962).

26. A. C. Hindmarsh, GEAR : Ordinary Differential Equation Solver UCID-30001 Rev.3, Lawrence Livermore Laboratory

(1974).

27. T. E. Hull, W. H. Enright, B. M. Fellen and A. E. Sedgwick, Comparing Numerical Methods for Ordinary Differential Equations, SIAM. J.Numer. Anal. 9, 603-637 (1972).

28. P. Kaps, Modifizierte Rosenbrock Methoden der Ordnung 4, 5 und 6 zur Numerischen Integration Steifer Differentialgleichungen, Thesis, University of Innsbruck (1977).

29. P. Kaps and P. Rentrop, Generalized Runge Kutta Methods of Order Four with Step Size Control for Stiff ODE's, Numer. Math. 33, 55-68 (1979).

30. P. Kaps and G. Wanner, A Study of Rosenbrock-type Methods of High Order, Submitted to Numerische Mathematik, (1979).

31. J. D. Lambert, Computational Methods in Ordinary Differential Equations, J. Wiley and Sons, New York (1973).

32. J. D. Lawson and B. L. Ehle, Improved Generalized Runge-Kutta, Proceedings of Canadian Computer Conference, Session 72, 223201-223213 (1972).

33. S. P. Norsett, Semi-Implicit Runge-Kutta Methods, Tech. Rept. Math. and Comp. No. 6/74, University of Trondheim (1974).

34. S. Poon, P. Kaps and T. D. Bui, A Comparison of ROW Methods for Stiff O.D.E.'s Using Richardson Extrapolation and Embedding Techniques for Stepsize Control, Dept. Comp. Sci., Concordia University, Montreal, (1980).

35. H. H. Rosenbrock, Some General Implicit Processes for The Numerical Solution of Differential Equations, Comput. J., 5, 329-330 (1963).

36. H. J. Stetter, Private Communications (1980).

37. G. Wanner, Letter to S.P. Norsett and Unpublished Communications to P.Kaps and A.Wolfbrandt, (1973).

38. G. Wanner, Private Communications (1978).

39. G. Wanner, On the Choice of γ for Singly-Explicit RK or Rosenbrock Methods, BIT 20, 102-106 (1980).

40. H. A. Watts, Survey of Numerical Methods for Ordinary Differential Equations, SAND79-2476, Sandia National

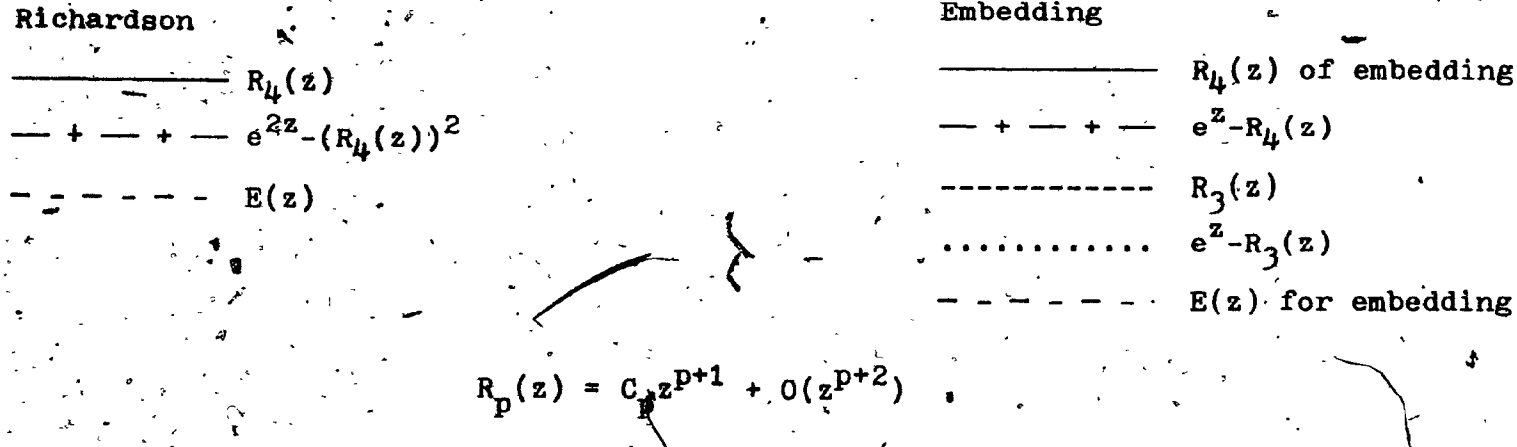
Laboratories (1980).

41. O. B. Widlund, A Note on Unconditionally Stable Linear Multistep Methods, BIT 7, 65-70 (1967).

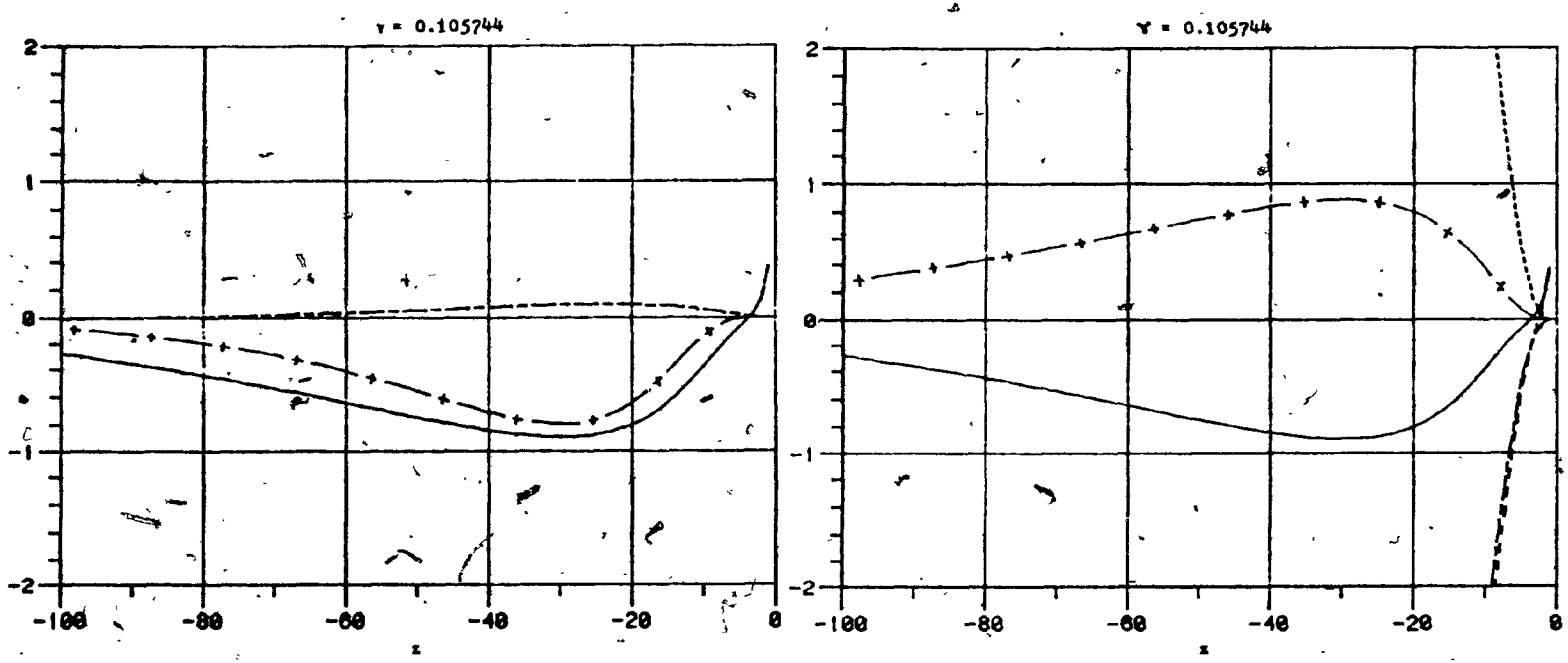
42. A. Wolfbrandt, A Study of Rosenbrock Processes with Respect to Order Conditions and Stiff Stability, Dissertation, Research Rep. 77.01 R, University of Goteborg, March (1977).

43. L. F. Shampine, Evaluation of Test Set for Stiff ODE Solvers, SAND80-2774, Sandia National Laboratories (1981).

APPENDIX A



$$R_p(z) = C_{p+1} z^{p+1} + O(z^{p+2})$$



$R_4(\infty) = 0.8914$

$C_4 \approx 3.2 \times 10^{-4}$

Small estimated error, but

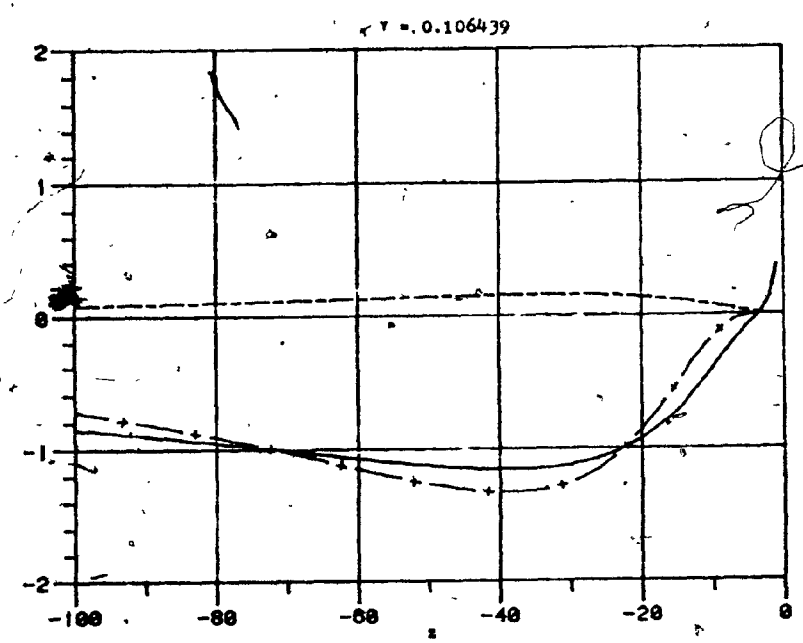
large actual and wrong sign

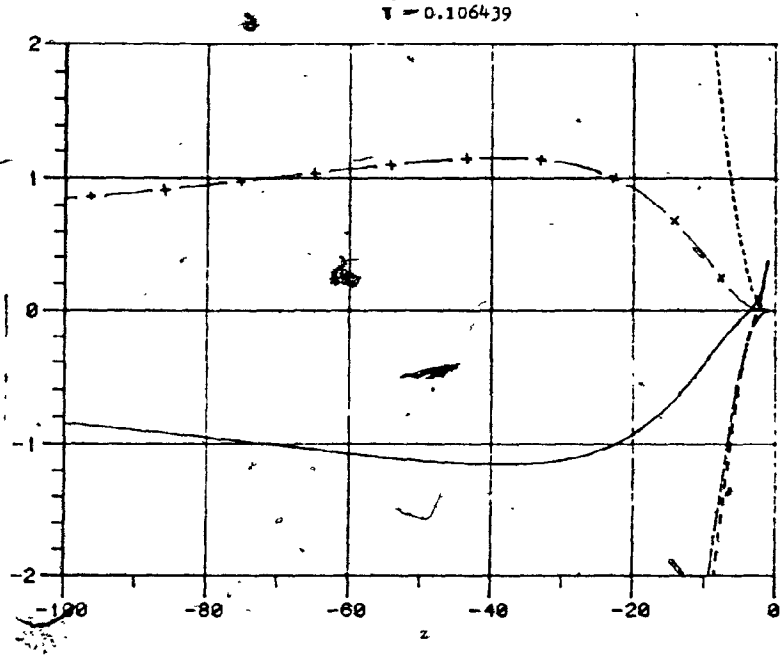
$R_3(\infty) = 34.1791$

$C_3 \approx 6.0 \times 10^{-3}$

Third order method unstable

no embedded method possible





$R_4(\infty) = 0.00$

$C_4 \approx 3.5 \times 10^{-4}$

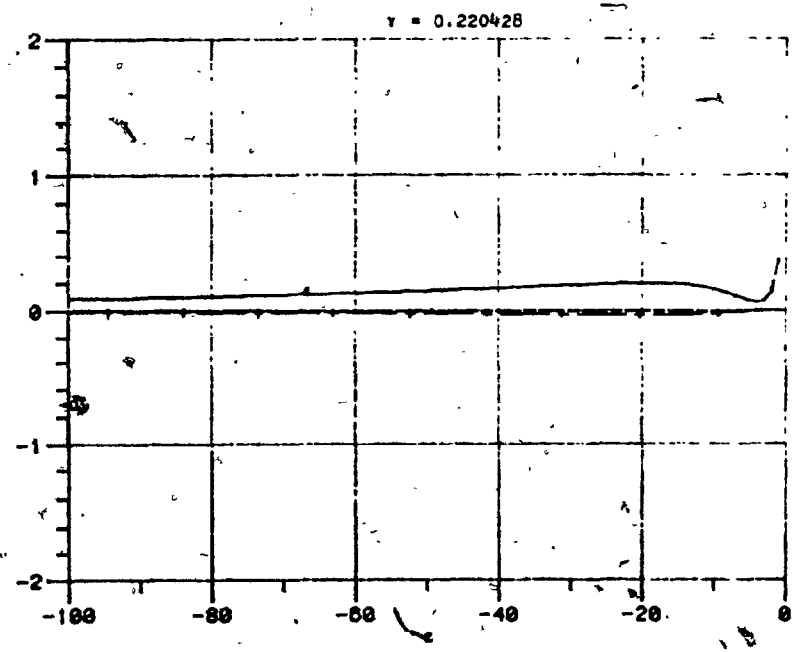
Not A(0)-stable, no ROWR4 method possible

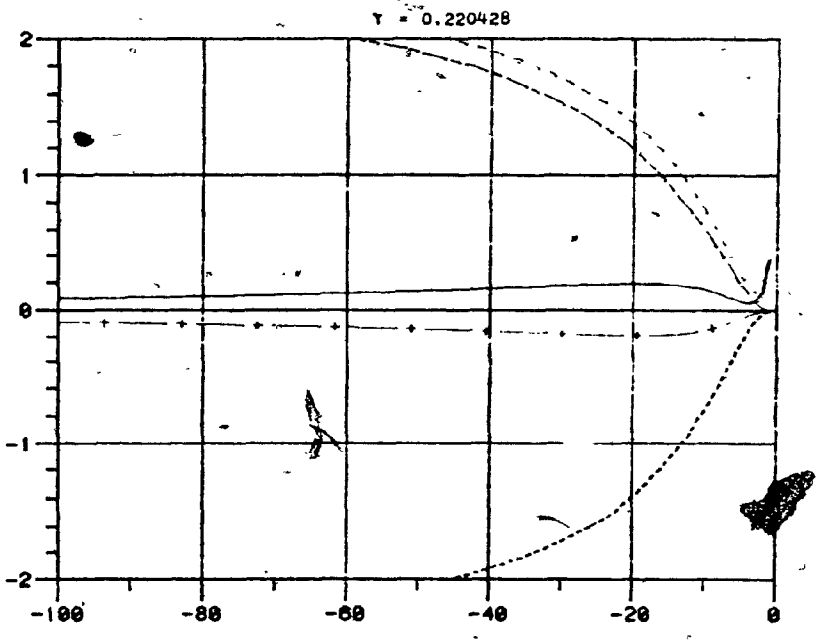
$R_3(\infty) = 32.9968$

$C_3 \approx 5.5 \times 10^{-3}$

third order method unstable

no embedded method possible





$R_4(\infty) = 0.00$

$C_4 \approx 1.1 \times 10^{-3}$

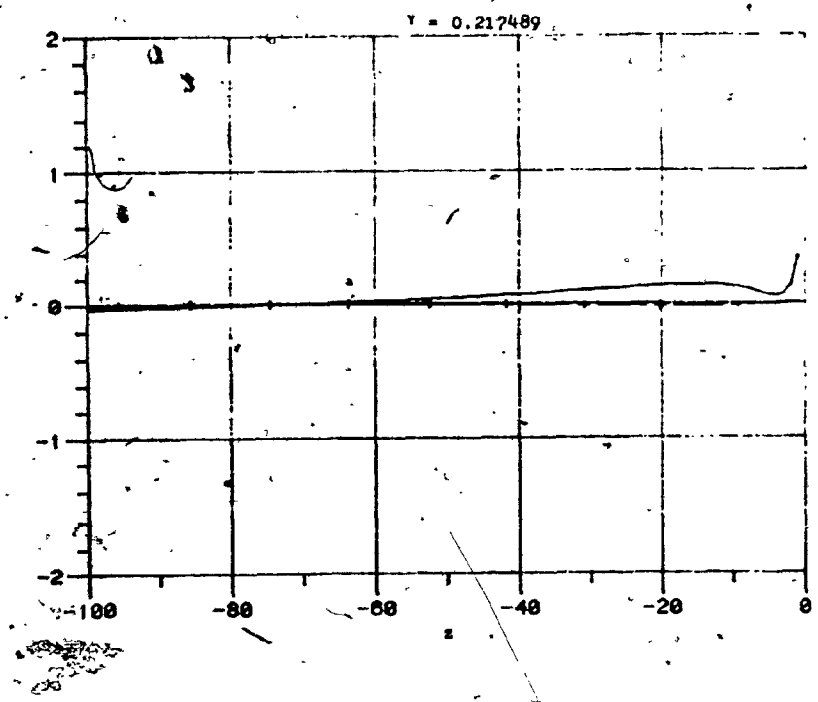
no change in sign,

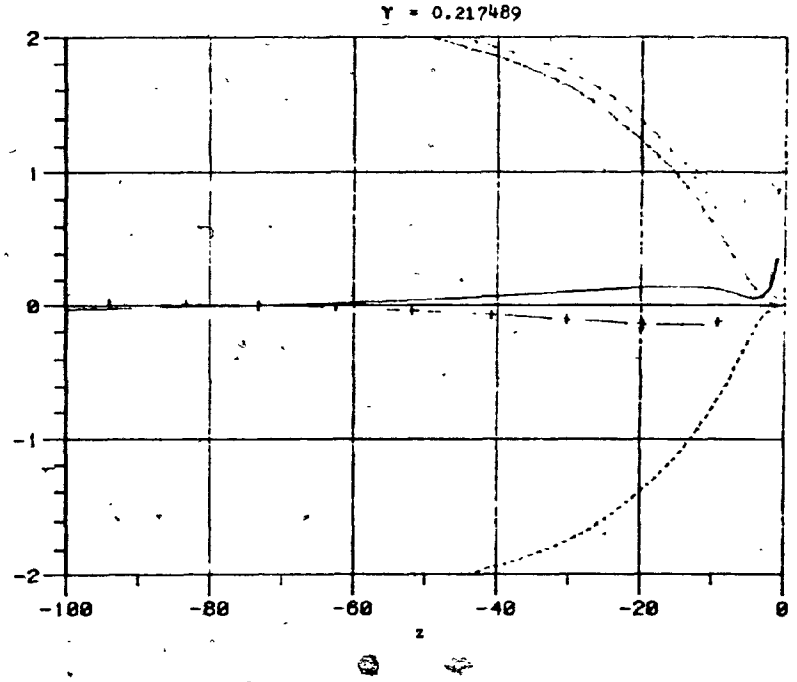
both actual and estimated error small

$R_3(\infty) = 2.7002$

$C_3 \approx 6.3 \times 10^{-3}$

Third order method is not A(0)-stable





$R_4(\infty) = 0.1494$

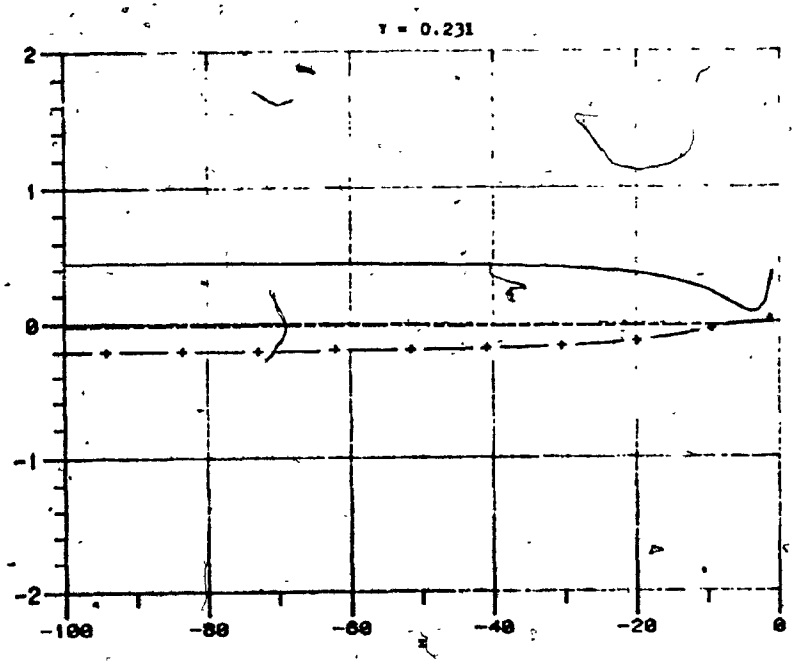
$C_4 \approx 1.0 \times 10^{-3}$

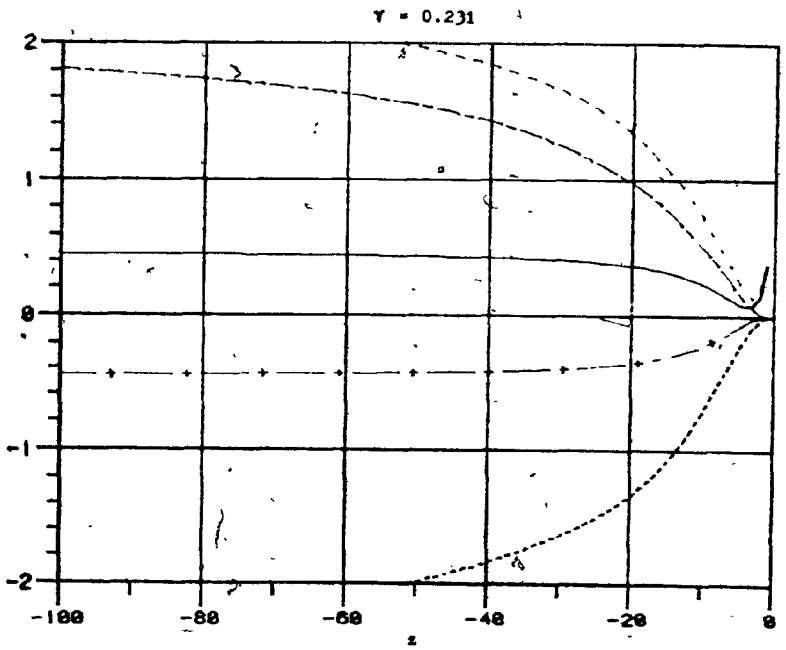
Chebyshev point, both actual and estimated error small

$R_3(\infty) = 2.7169$

$C_3 \approx 7.1 \times 10^{-3}$

Third order method unstable estimated error too big





$R_4(\infty) = 0.4536$

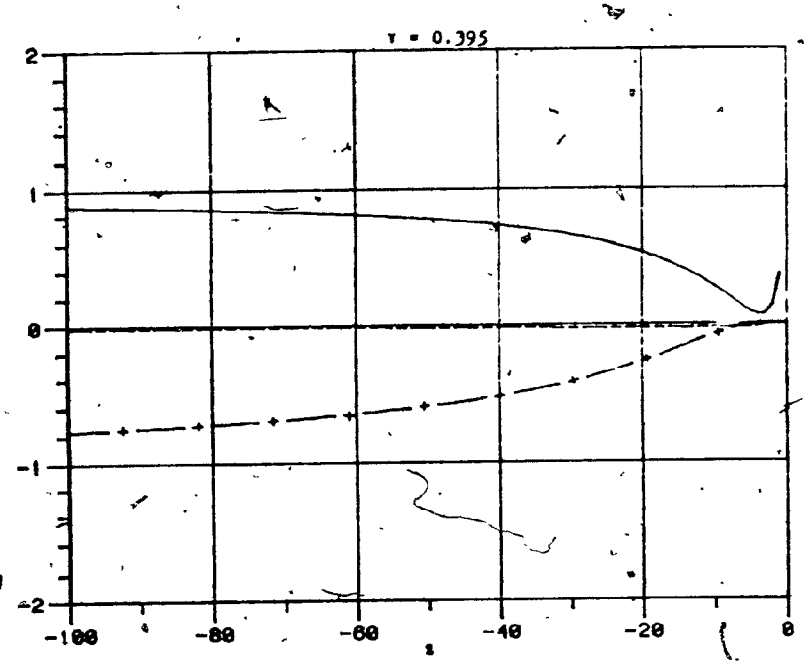
$C_4 \approx 1.2 \times 10^{-3}$

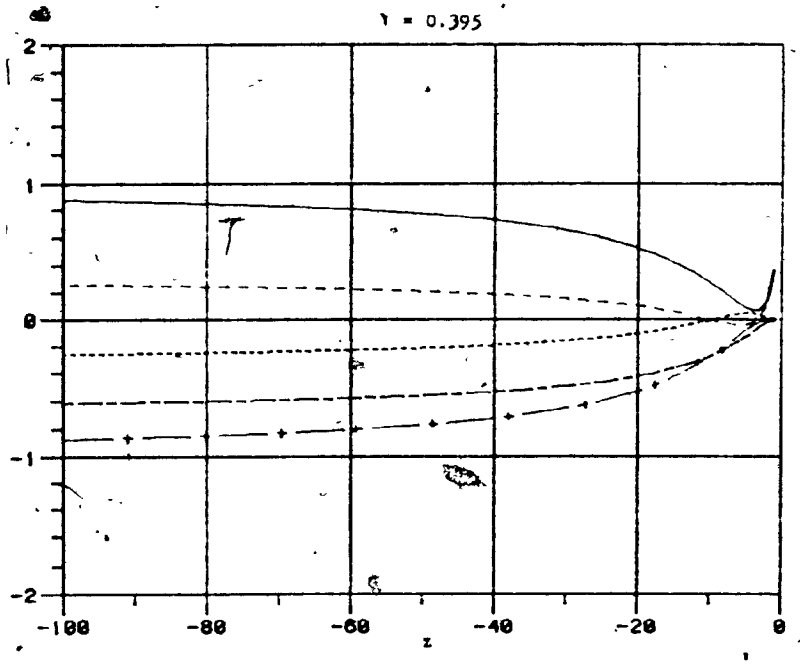
estimated error smaller than actual error

$R_3(\infty) = 2.6023$

$C_3 \approx 5.9 \times 10^{-3}$

Third order method is not A(0)-stable





$R_4(\infty) = 0.9954$

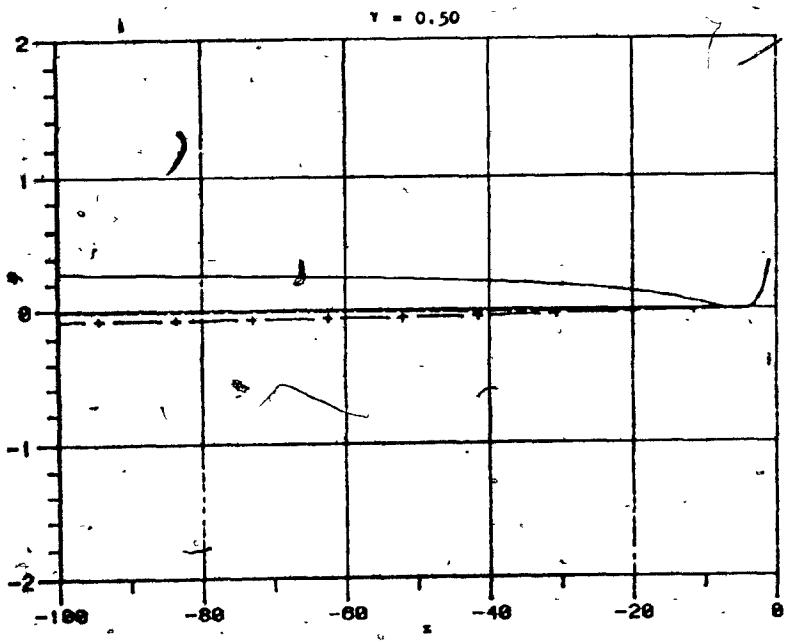
$C_4 \approx 1.6 \times 10^{-4}$

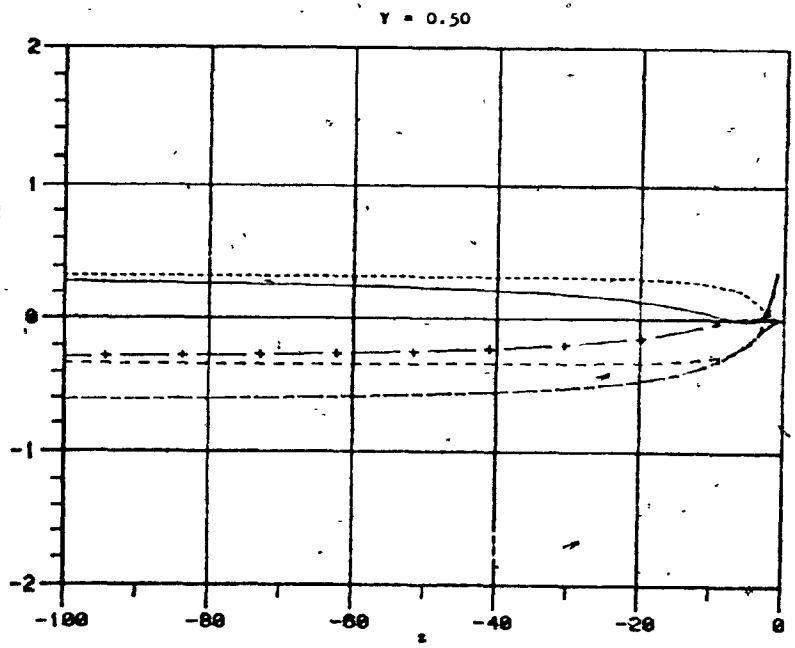
A-stable, estimated error is smaller than the actual error

$R_3(\infty) = 0.3146$

$C_3 \approx 1.6 \times 10^{-2}$

Both third and fourth order methods are A-stable





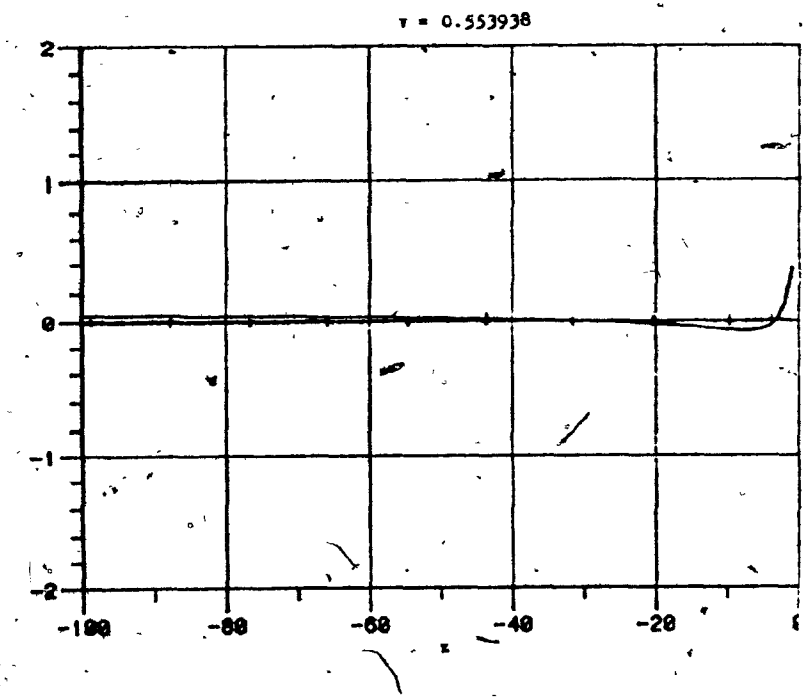
$R_4(\infty) = 0.3333$

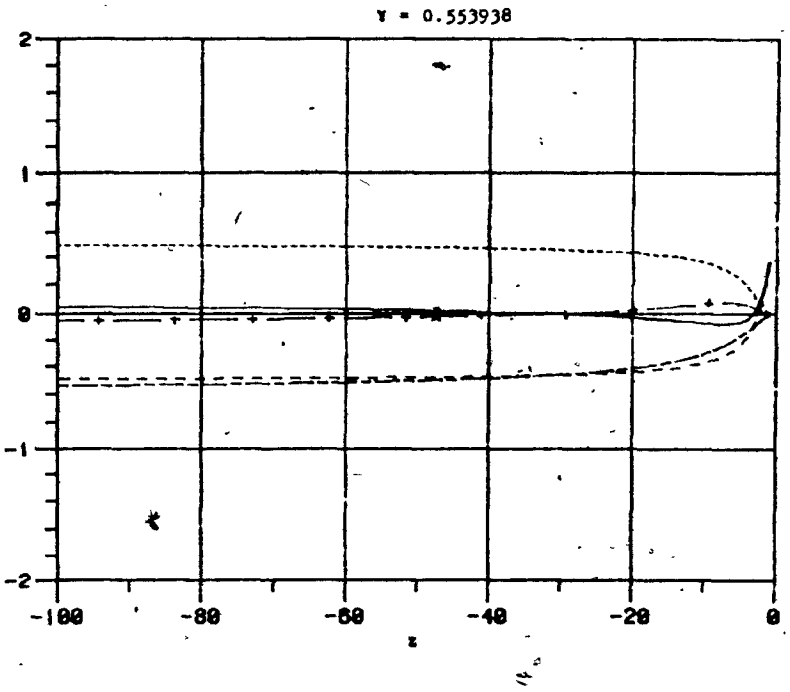
$C_4 \approx 1.2 \times 10^{-2}$

A-stable

$R_3(\infty) = 0.3333$

$C_3 \approx 4.0 \times 10^{-2}$





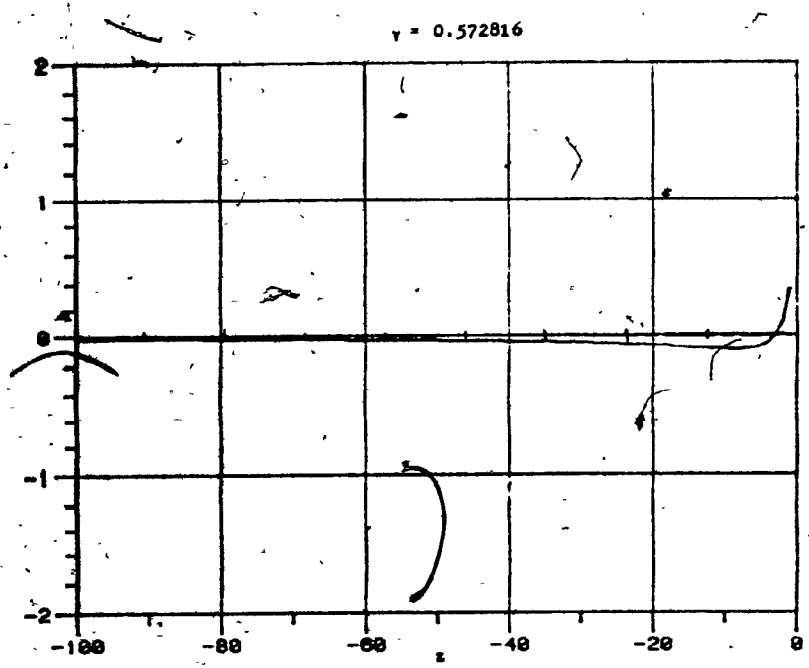
$R_4(\infty) = 0.0762$

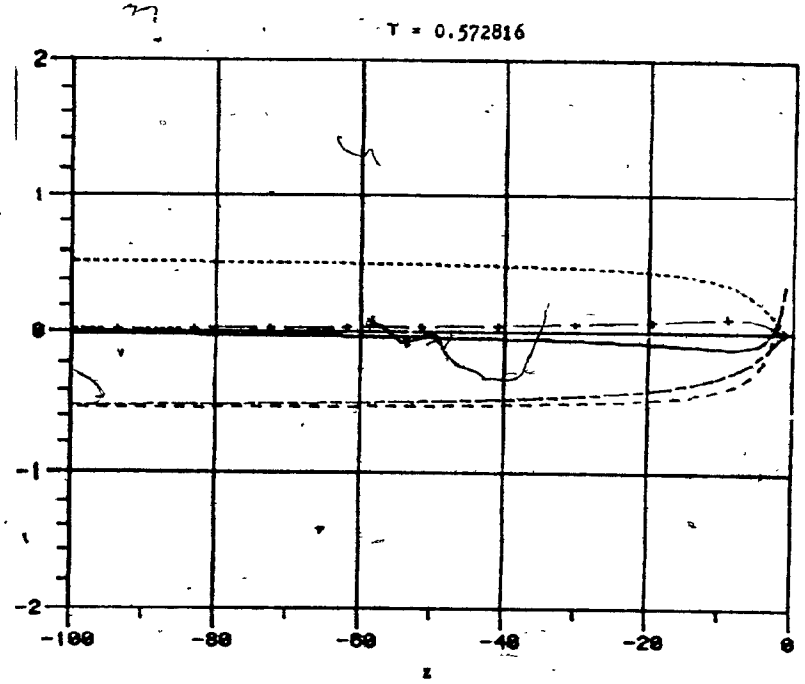
$C_4 \approx 2.4 \times 10^{-2}$

Chebyshev point, both actual and estimated error small, big local truncation error

$R_3(\infty) = 0.5079$

$C_3 \approx 5.6 \times 10^{-2}$





$R_4(\infty) = 0.00$

$C_4 \approx 3.2 \times 10^{-2}$

big local truncation error

$R_3(\infty) = 0.5525$

$C_3 \approx 6.0 \times 10^{-2}$

	SUMMARY	1.003	828	276	276	1644	254
10**-4	D1	.741	603	201	201	1206	199
	D2	.318	249	83	83	498	81
	D3	.222	138	44	46	312	42
	D4	.104	90	30	30	180	28
	D5	.143	162	54	54	270	38
	D6	.170	135	45	45	270	43
	SUMMARY	1.698	1377	457	459	2736	431

CLASS E ROW4A

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.071	36	12	12	84	10
	E2	.043	51	17	17	85	15
	E3	.109	87	29	29	174	27
	E4	.272	150	50	50	350	44
	E5	.207	126	42	42	294	40
	SUMMARY	.702	450	150	150	987	136
10**-3	E1	.074	39	13	13	91	11
	E2	.043	51	17	17	85	15
	E3	.151	123	40	41	242	38
	E4	.363	195	63	65	445	59
	E5	.203	126	42	42	294	40
	SUMMARY	.834	534	175	178	1157	163
10**-4	E1	.090	48	15	16	107	13
	E2	.045	57	19	19	95	17
	E3	.308	258	84	86	508	80
	E4	.514	285	91	95	645	87
	E5	.207	126	42	42	294	40
	SUMMARY	1.164	774	251	258	1649	237

CLASS F ROW4A

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.095	117	38	39	192	28
	F2	.084	102	34	34	170	26
	F3	1.872	1482	493	494	2960	423
	F4	.508	408	135	136	812	95
	F5	.480	120	40	40	440	32

	SUMMARY	3.039	2229	740	743	4574	604
10**-3	F1	.140	165	55	55	275	33
	F2	.123	144	48	48	240	38
	F3	2.034	1608	536	536	3216	486
	F4	.638	510	170	170	1020	138
	F5	.657	162	54	54	594	44
	SUMMARY	3.592	2589	863	863	5345	739
10**-4	F1	.170	198	66	66	330	44
	F2	.170	204	68	68	340	58
	F3	2.755	2226	741	742	4448	713
	F4	.912	732	244	244	1464	228
	F5	.749	186	62	62	682	50
	SUMMARY	4.756	3546	1181	1182	7264	1093

CLASS A ROWR4(FORMULA 3.13)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	A1	.099	96	16	24	152	12
	A2	.413	132	22	33	319	18
	A3	.169	144	24	36	228	20
	A4	.618	168	28	42	434	24
	SUMMARY	1.299	540	90	135	1133	74
10**-3	A1	.120	120	20	30	190	16
	A2	.485	156	26	39	377	22
	A3	.244	216	36	54	342	26
	A4	.791	216	36	54	558	32
	SUMMARY	1.640	708	118	177	1467	96
10**-4	A1	.198	192	32	48	304	24
	A2	.706	228	38	57	551	30
	A3	.331	288	48	72	456	36
	A4	1.157	312	52	78	806	44
	SUMMARY	2.392	1020	170	255	2117	134

CLASS B ROWR4(FORMULA 3.13)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	B1	.477	408	68	102	646	46
	B2	.165	96	16	24	184	12
	B3	.158	96	16	24	184	12
	B4	.212	120	20	30	230	16
	B5	.402	228	38	57	437	34
	SUMMARY	1.414	948	158	237	1681	120
10**-3	B1	.917	840	140	210	1330	86
	B2	.241	144	24	36	276	16
	B3	.260	156	26	39	299	18
	B4	.326	192	32	48	368	24
	B5	.687	396	66	99	759	58
	SUMMARY	2.431	1728	288	432	3032	202
10**-4	B1	1.646	1512	252	378	2394	150
	B2	.313	180	30	45	345	24
	B3	.337	192	32	48	368	26
	B4	.460	264	44	66	506	38
	B5	1.197	684	114	171	1311	98

	SUMMARY	3.953	2832	472	708	4924	336

CLASS C ROWR4(FORMULA 3.13)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.140	120	20	30	190	16
	C2	.129	108	18	27	171	14
	C3	.126	108	18	27	171	14
	C4	.164	132	22	33	209	18
	C5	.166	132	22	33	209	18
	SUMMARY	.725	600	100	150	950	80
10**-3	C1	.174	156	26	39	247	20
	C2	.178	144	24	36	228	20
	C3	.170	144	24	36	228	20
	C4	.218	180	30	45	285	25
	C5	.235	192	32	48	304	28
	SUMMARY	.975	816	136	204	1292	114
10**-4	C1	.246	216	36	54	342	30
	C2	.249	216	36	54	342	28
	C3	.264	216	36	54	342	28
	C4	.309	252	42	63	399	38
	C5	.377	300	50	75	475	46
	SUMMARY	1.445	1200	200	300	1900	170

CLASS D ROWR4(FORMULA 3.13)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.170	204	34	51	289	12
	D2	.126	144	24	36	204	18
	D3	.164	144	24	36	228	20
	D4	.057	72	12	18	102	8
	D5	.071	120	20	30	150	12
	D6	.068	84	14	21	119	10
	SUMMARY	.656	768	128	192	1092	80
10**-3	D1	.179	216	36	54	306	20
	D2	.163	192	32	48	272	22
	D3	.188	168	28	42	266	24
	D4	.057	72	12	18	102	8
	D5	.062	108	18	27	135	12
	D6	.077	96	16	24	136	12

	SUMMARY	.726	852	142	213	1217	98
10**-4	D1	.289	336	56	84	476	32
	D2	.190	216	36	54	306	30
	D3	.248	216	36	54	342	32
	D4	.077	96	16	24	136	10
	D5	.122	204	34	51	255	18
	D6	.099	120	20	30	170	16
	SUMMARY	1.025	1188	198	297	1685	138

CLASS E ROWR4(FORMULA 3.13)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.084	60	10	15	95	6
	E2	.026	48	8	12	60	4
	E3	.109	132	22	33	187	16
	E4	.390	300	50	75	475	36
	E5	.076	72	12	18	114	6
	SUMMARY	.685	612	102	153	931	68
10**-3	E1	.097	72	12	18	114	8
	E2	.061	108	18	27	135	6
	E3	.118	144	24	36	204	20
	E4	.508	396	66	99	627	46
	E5	.076	72	12	18	114	6
	SUMMARY	.860	792	132	198	1194	86
10**-4	E1	.115	84	14	21	133	10
	E2	.062	108	18	27	135	8
	E3	.189	228	38	57	323	30
	E4	.721	564	94	141	893	62
	E5	.092	84	14	21	133	8
	SUMMARY	1.179	1068	178	267	1617	118

CLASS F ROWR4(FORMULA 3.13)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.113	192	32	48	240	18
	F2	.379	636	106	159	795	68
	F3	1.985	2256	376	564	3196	182
	F4	.740	864	144	216	1224	76
	F5	1.091	348	58	121	783	42

	SUMMARY	4.308	4296	716	1108	6238	386
10**-3	F1	.104	180	30	45	225	18
	F2	.212	360	60	90	450	32
	F3	1.018	1200	200	300	1700	126
	F4	.628	720	120	180	1020	76
	F5	.634	228	38	57	513	26
	SUMMARY	2.596	2688	448	672	3908	278
10**-4	F1	.210	360	60	90	450	32
	F2	.197	336	56	84	420	36
	F3	1.674	1968	328	492	2788	190
	F4	.829	960	160	240	1360	104
	F5	1.078	384	64	96	864	40
	SUMMARY	3.988	4008	668	1002	5882	402

CLASS A GEAR

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	A1	.113	-	12	12	107	41
	A2	.264	-	14	14	188	47
	A3	.143	-	18	18	262	63
	A4	.385	-	18	18	262	63
	SUMMARY	.905	-	62	62	819	214
10**-3	A1	.179	-	16	16	153	65
	A2	.376	-	17	17	244	70
	A3	.222	-	17	17	180	82
	A4	.632	-	21	21	336	101
	SUMMARY	1.409	-	71	71	913	318
10**-4	A1	.252	-	16	16	188	91
	A2	.594	-	23	23	346	108
	A3	.319	-	21	21	238	122
	A4	.856	-	26	26	431	140
	SUMMARY	2.021	-	86	86	1203	461

CLASS B GEAR

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	B1	.350	-	18	18	269	158
	B2	.127	-	9	9	106	36
	B3	.139	-	10	10	115	38
	B4	.205	-	13	13	155	56
	B5	6.616	-	121	121	3148	2356
	SUMMARY	7.437	-	171	171	3793	2644
10**-3	B1	.822	-	27	27	496	311
	B2	.225	-	12	12	149	56
	B3	.219	-	14	14	164	61
	B4	.296	-	14	14	197	83
	B5	6.782	-	125	125	3219	2399
	SUMMARY	8.344	-	192	192	4225	2910
10**-4	B1	.935	-	29	29	600	376
	B2	.280	-	14	14	187	77
	B3	.304	-	14	14	194	83
	B4	1.411	-	34	34	705	405
	B5	7.750	-	134	134	3413	2311

	SUMMARY	10.680	-	225	225	5099	3252

CLASS C GEAR

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.013	-	13	13	116	47
	C2	.117	-	13	13	110	43
	C3	.124	-	13	13	114	45
	C4	.138	-	13	13	121	49
	C5	.130	-	12	12	117	44
	SUMMARY	.522	-	64	64	578	228
10**-3	C1	.222	-	18	18	175	76
	C2	.211	-	15	15	155	71
	C3	.218	-	15	15	157	72
	C4	.216	-	17	17	177	77
	C5	.202	-	17	17	170	72
	SUMMARY	1.069	-	82	82	834	368
10**-4	C1	.293	-	18	18	207	103
	C2	.267	-	18	18	199	94
	C3	.275	-	18	18	200	97
	C4	.284	-	19	19	209	101
	C5	.297	-	18	18	218	108
	SUMMARY	1.416	-	91	91	1033	503

CLASS D GEAR

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.084	-	10	10	114	29
	D2	.088	-	10	10	84	40
	D3	.142	-	17	17	144	54
	D4	.034	-	5	5	30	14
	D5	.041	-	9	9	47	22
	D6	.047	-	7	7	48	21
	SUMMARY	.436	-	58	58	467	180
10**-3	D1	.121	-	11	11	134	43
	D2	.130	-	13	13	121	59
	D3	.214	-	22	22	193	84
	D4	.039	-	5	5	34	16
	D5	.075	-	15	15	92	34
	D6	.060	-	7	7	59	28

	SUMMARY	.639	-	73	73	633	264
10**-4	D1	.170	-	12	12	186	66
	D2	.207	-	19	19	186	94
	D3	.335	-	23	23	260	128
	D4	.046	-	6	6	41	20
	D5	.103	-	17	17	143	51
	D6	.085	-	8	8	78	40
	SUMMARY	.946	-	85	85	894	399

CLASS E GEAR

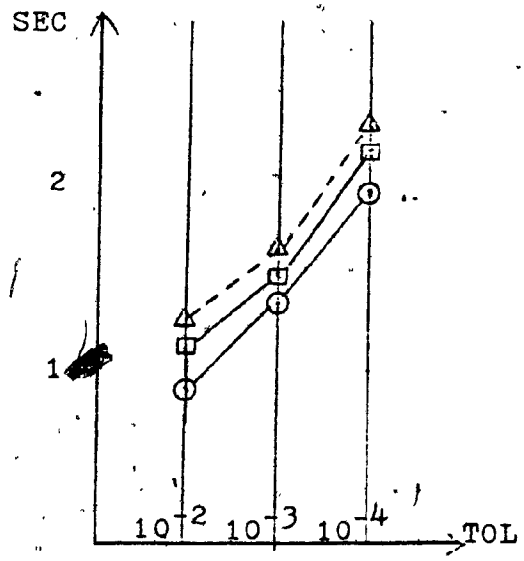
TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.027	-	3	3	21	7
	E2	.028	-	4	4	27	14
	E3	.100	-	10	10	101	43
	E4	.216	-	19	19	201	71
	E5	.051	-	5	5	38	17
	SUMMARY	.422	-	41	41	388	152
10**-3	E1	.074	-	8	8	60	18
	E2	.041	-	6	6	45	22
	E3	.135	-	13	13	124	57
	E4	.360	-	25	25	276	120
	E5	.050	-	6	6	44	18
	SUMMARY	.660	-	58	58	549	235
10**-4	E1	.098	-	9	9	84	27
	E2	.058	-	6	6	65	33
	E3	.204	-	16	16	183	94
	E4	.504	-	32	32	357	171
	E5	.066	-	5	5	52	23
	SUMMARY	.930	-	68	68	741	348

CLASS F GEAR

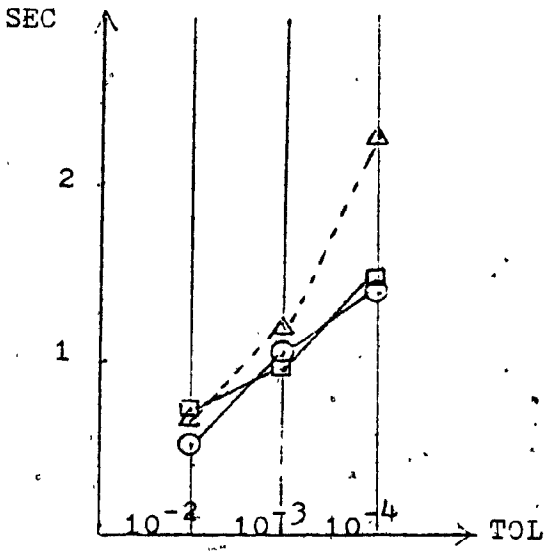
TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.158	-	30	30	230	73
	F2	.072	-	12	12	95	39
	F3	.297	-	34	34	360	119
	F4	.052	-	7	7	54	24
	F5	.341	-	19	19	254	51

	SUMMARY	.920	-	102	102	993	306
10**-3	F1	.091	-	17	17	126	42
	F2	.108	-	12	12	125	58
	F3	.509	-	49	49	562	206
	F4	.082	-	8	8	83	38
	F5	.641	-	30	30	445	101
	SUMMARY	1.431	-	116	116	1341	445
10**-4	F1	.121	-	14	14	163	57
	F2	.144	-	10	10	143	79
	F3	.678	-	46	46	644	277
	F4	.187	-	13	13	152	80
	F5	.936	-	31	31	540	159
	SUMMARY	2.066	-	114	114	1642	652

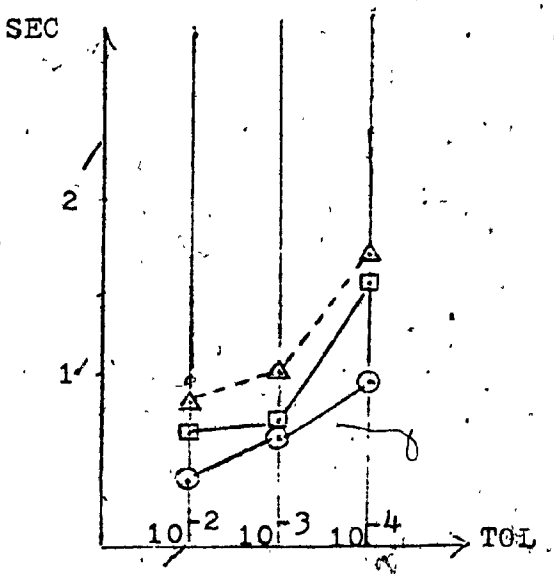
The following graphs show the computing time of ROW4A, ROWR4 and GEAR over the six different choices of problems.



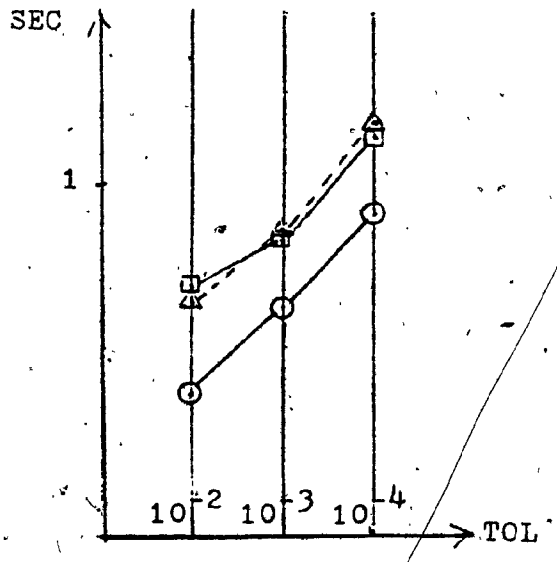
Class A
Linear with real eigenvalues



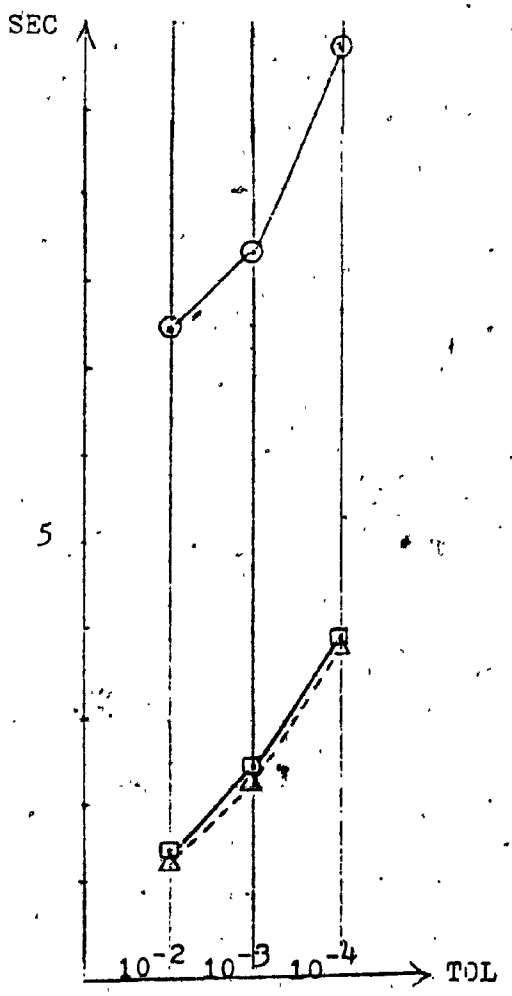
Class B
Non-linear coupling



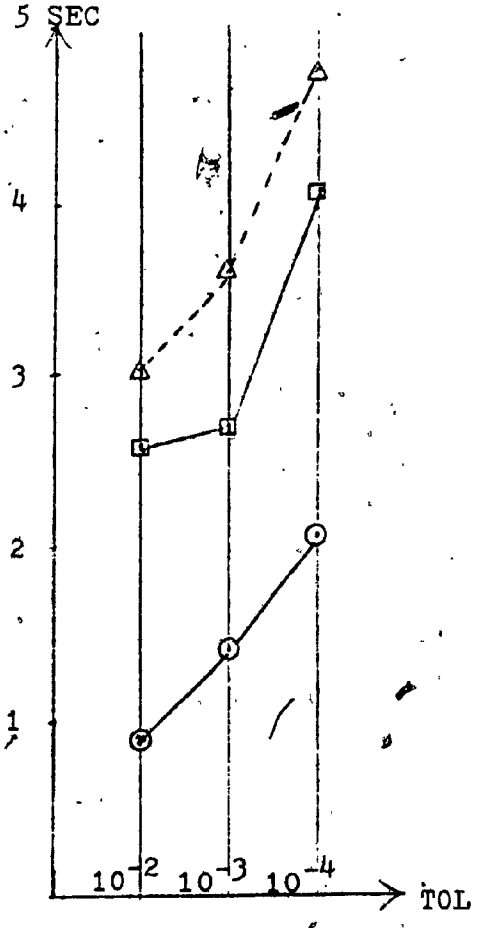
Class D
Nonlinear with real eigenvalues



Class E
Nonlinear with complex eigenvalues



Class B
Linear with complex eigenvalues



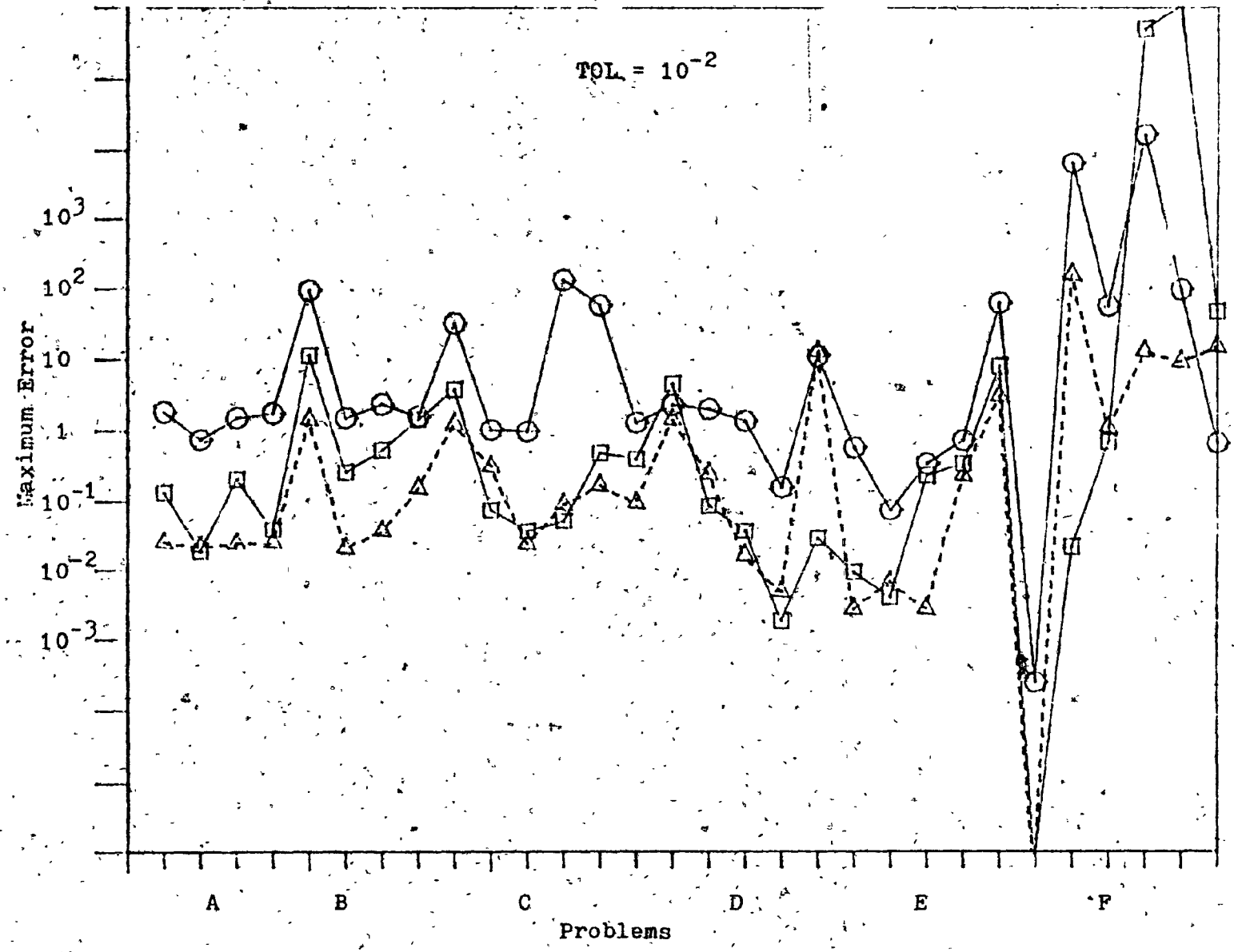
Class F

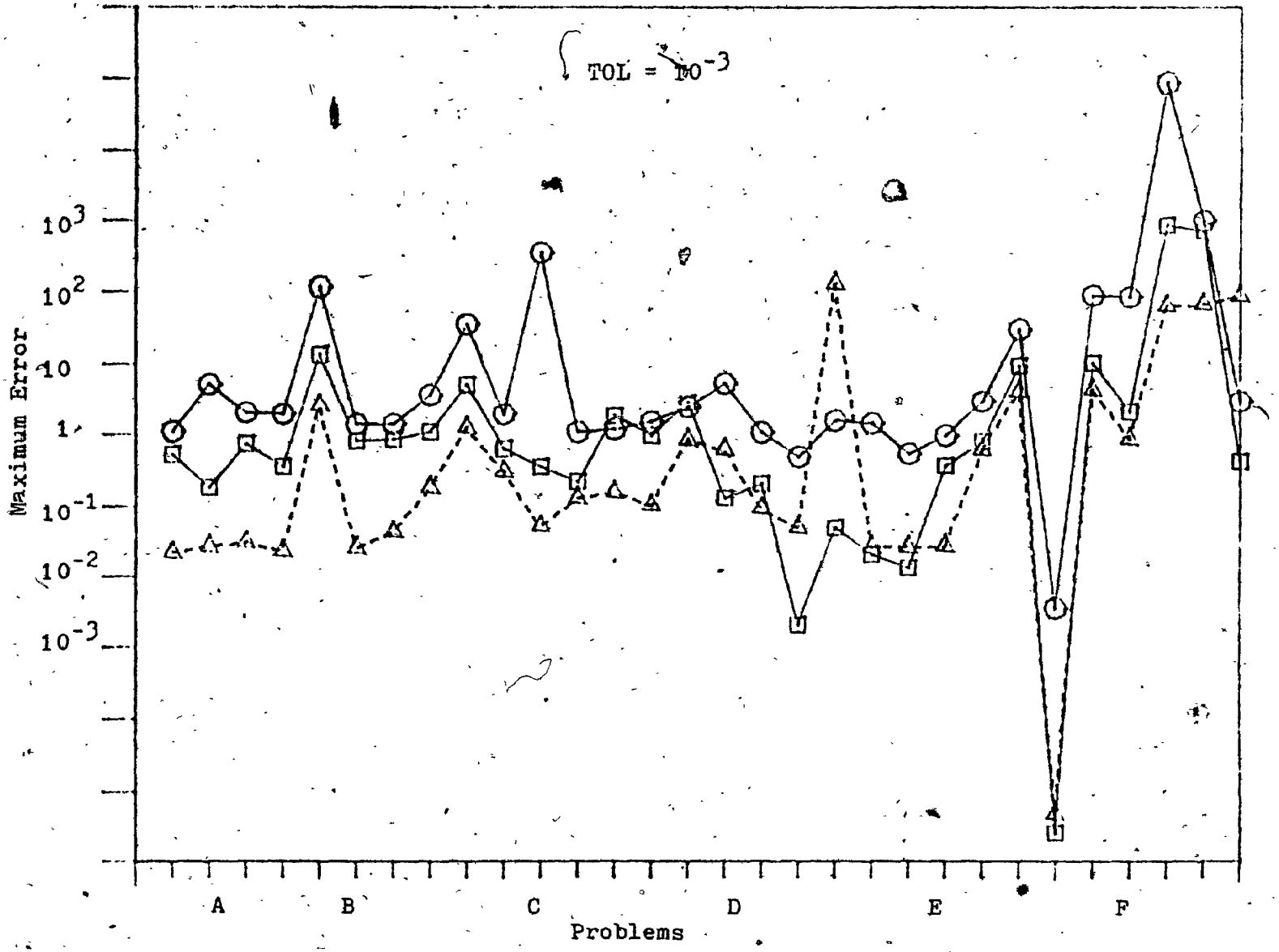
GEAR

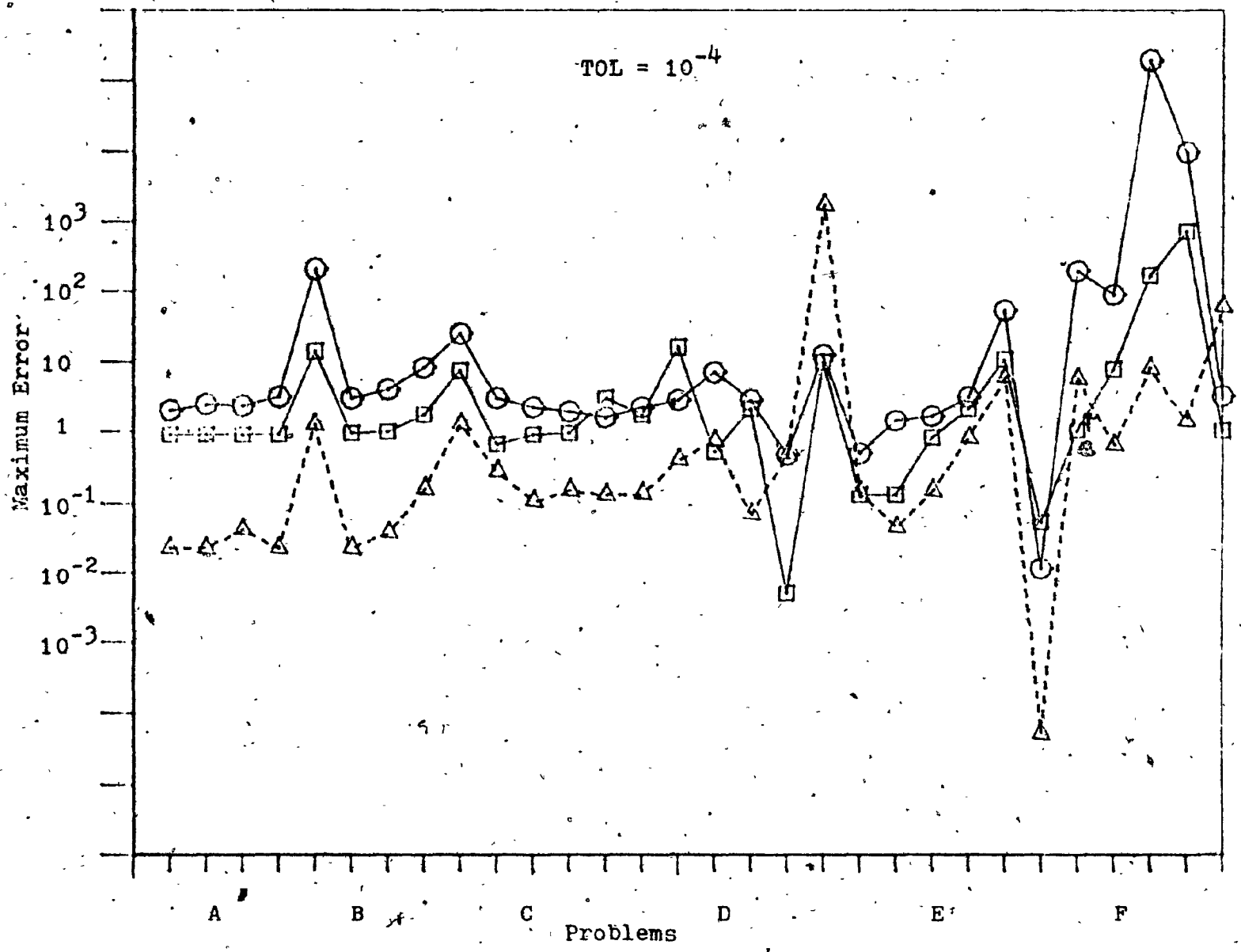
ROWR4(formula 3.13)

ROW4A

In the following figure the maximal global errors are plotted for three tolerance TOL=10^{-2}, 10^{-3}, 10^{-4}. For each component with an absolute value greater than one, we used the relative error. For those components with absolute value smaller than 1, we used the absolute error. The exact solution were obtained by GEAR with TOL=10^{-10}. The graphs show for each example, the maximum of these errors, over the whole integration interval and all components divided by TOL. The logarithmic scale was used.


TOL = 10^{-2}
Maximum Error
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems


TOL = 10^-3
Maximum Error
10^3
10^2
10
1
10^-1
10^-2
10^-3
A
B
C
D
E
F
Problems


TOL = 10^{-4}
Maximum Error
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems

	SUMMARY	5.838	3462	1154	1154	9190	876

CLASS C ROB3A(FORMULA 4.8)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.150	114	38	38	266	26
	C2	.130	96	32	32	224	22
	C3	.132	90	30	30	210	24
	C4	.205	138	46	46	322	42
	C5	.331	222	74	74	518	58
	SUMMARY	.948	660	220	220	1540	172
10**-3	C1	.227	156	52	52	364	42
	C2	.207	150	50	50	350	38
	C3	.228	162	54	54	378	42
	C4	.449	300	100	100	700	88
	C5	.619	408	136	136	952	122
	SUMMARY	1.730	1176	392	392	2744	332
10**-4	C1	.338	240	80	80	560	70
	C2	.319	222	74	74	518	64
	C3	.399	270	90	90	630	80
	C4	.978	642	214	214	1498	192
	C5	1.197	792	264	264	1848	252
	SUMMARY	3.231	2166	722	722	5054	658

CLASS D ROB3A(FORMULA 4.8)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.382	390	130	130	780	40
	D2	.112	114	38	38	228	28
	D3	.137	102	34	34	238	30
	D4	.037	42	14	14	84	10
	D5	.068	96	32	32	160	18
	D6	.045	48	16	16	96	12
	SUMMARY	.781	792	264	264	1586	138
10**-3	D1	.862	840	280	280	1680	110
	D2	.161	162	54	54	324	42
	D3	.223	168	56	56	392	44
	D4	.045	48	16	16	96	12
	D5	.068	102	34	34	170	20
	D6	.060	66	22	22	132	18

	SUMMARY	1.419	1386	462	462	2794	246
10**-4	D1	1.409	1416	472	472	2832	316
	D2	.331	306	102	102	612	90
	D3	.320	234	78	78	546	68
	D4	.051	54	18	18	108	14
	D5	.116	162	54	54	270	30
	D6	.092	90	30	30	180	24
	SUMMARY	2.319	2262	754	754	4548	542

CLASS E ROB3A(FORMULA 4.8)

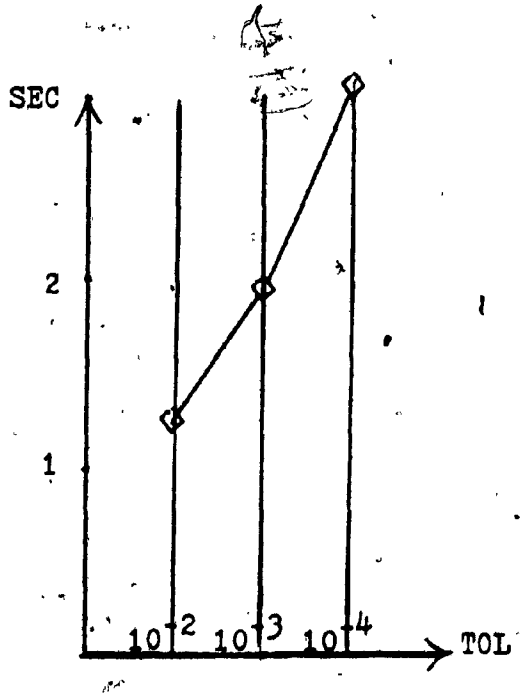
TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.061	36	12	12	84	8
	E2	.023	36	12	12	60	8
	E3	.111	114	38	38	228	28
	E4	.330	210	70	70	490	56
	E5	.046	36	12	12	84	6
	SUMMARY	.571	432	144	144	946	106
10**-3	E1	.067	42	14	14	98	10
	E2	.031	48	16	16	80	12
	E3	.259	258	86	86	516	56
	E4	.550	348	116	116	812	88
	E5	.051	42	14	14	98	8
	SUMMARY	.958	738	246	246	1604	174
10**-4	E1	.155	90	30	30	210	16
	E2	.051	78	26	26	130	18
	E3	.500	516	172	172	1032	128
	E4	.870	564	188	188	1316	146
	E5	.046	36	12	12	84	8
	SUMMARY	1.622	1284	428	428	2772	316

CLASS F ROB3A(FORMULA 4.8)

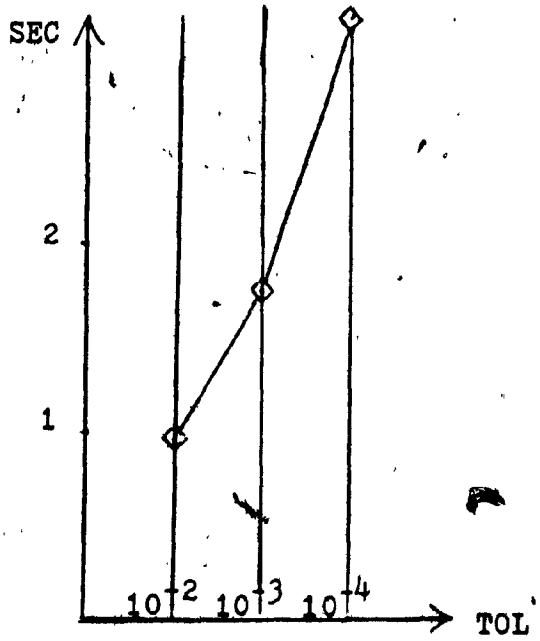
TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.064	96	32	32	160	18
	F2	.113	162	54	54	270	32
	F3	1.081	1056	352	352	2112	166
	F4	.492	498	166	166	996	94
	F5	.253	78	26	26	286	22

	SUMMARY	2.003	1890	630	630	3824	332
10**-3	F1	.129	192	64	64	320	28
	F2	.162	234	78	78	390	50
	F3	2.202	2160	720	720	4320	336
	F4	1.017	996	332	332	1992	172
	F5	.602	174	58	58	638	42
	SUMMARY	4.112	3756	1252	1252	7660	628
10**-4	F1	.347	492	164	164	820	56
	F2	.264	396	132	132	660	86
	F3	4.343	4320	1440	1440	8640	772
	F4	1.969	1956	652	652	3912	354
	F5	.999	288	96	96	1056	72
	SUMMARY	7.922	7452	2484	2484	15088	1340

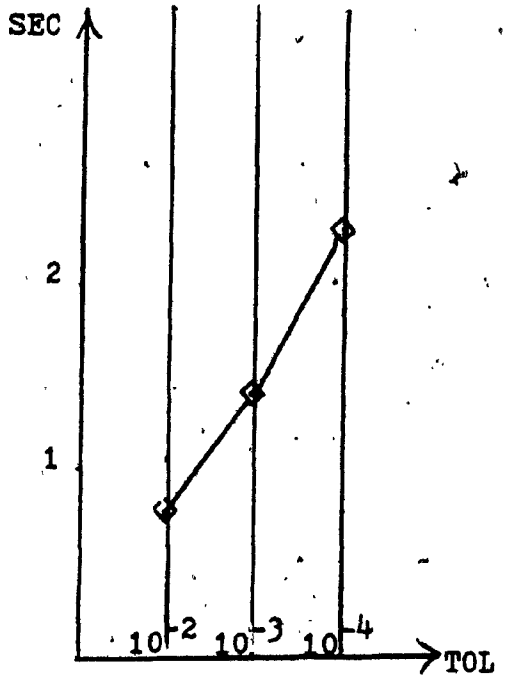
The following graphs show the computing time of ROB3A over the six different classes of problems.



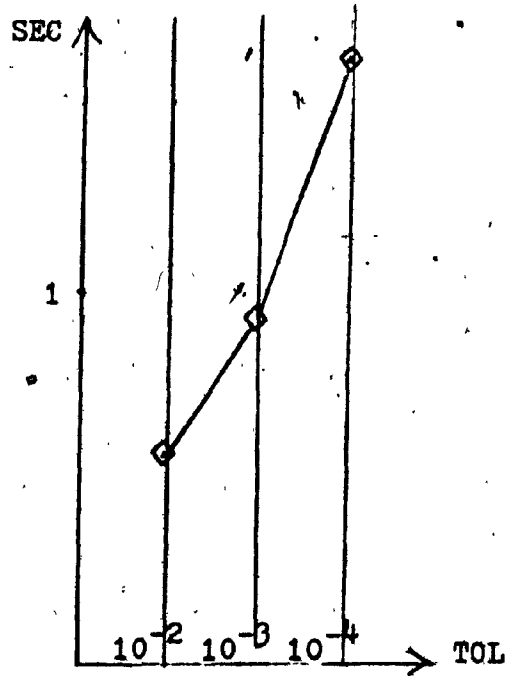
Class A
Linear with real eigenvalues



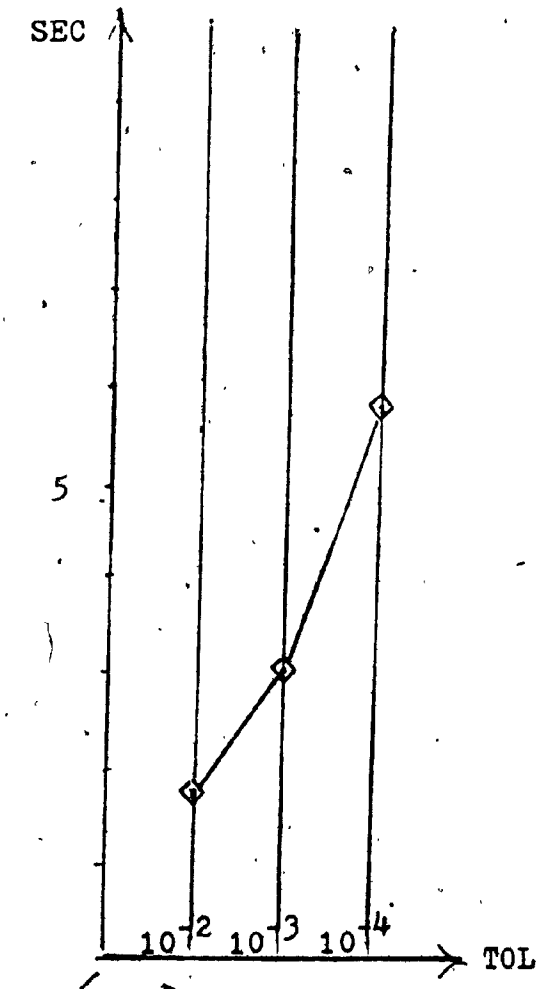
Class C
Non-linear Coupling



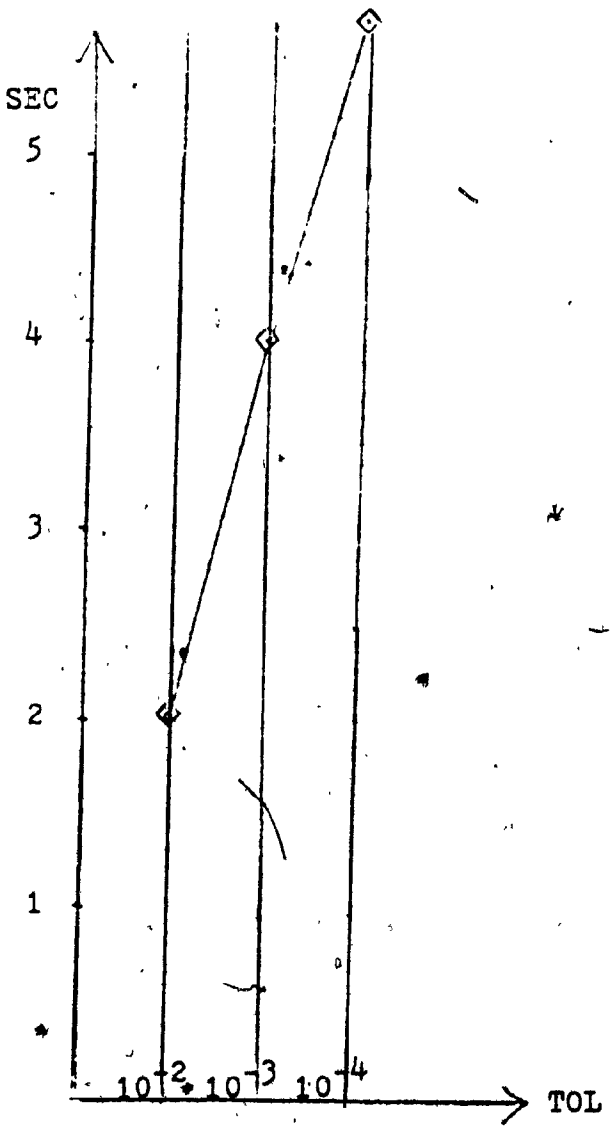
Class D
Nonlinear with real eigenvalues



Class E
Nonlinear with complex eigenvalues



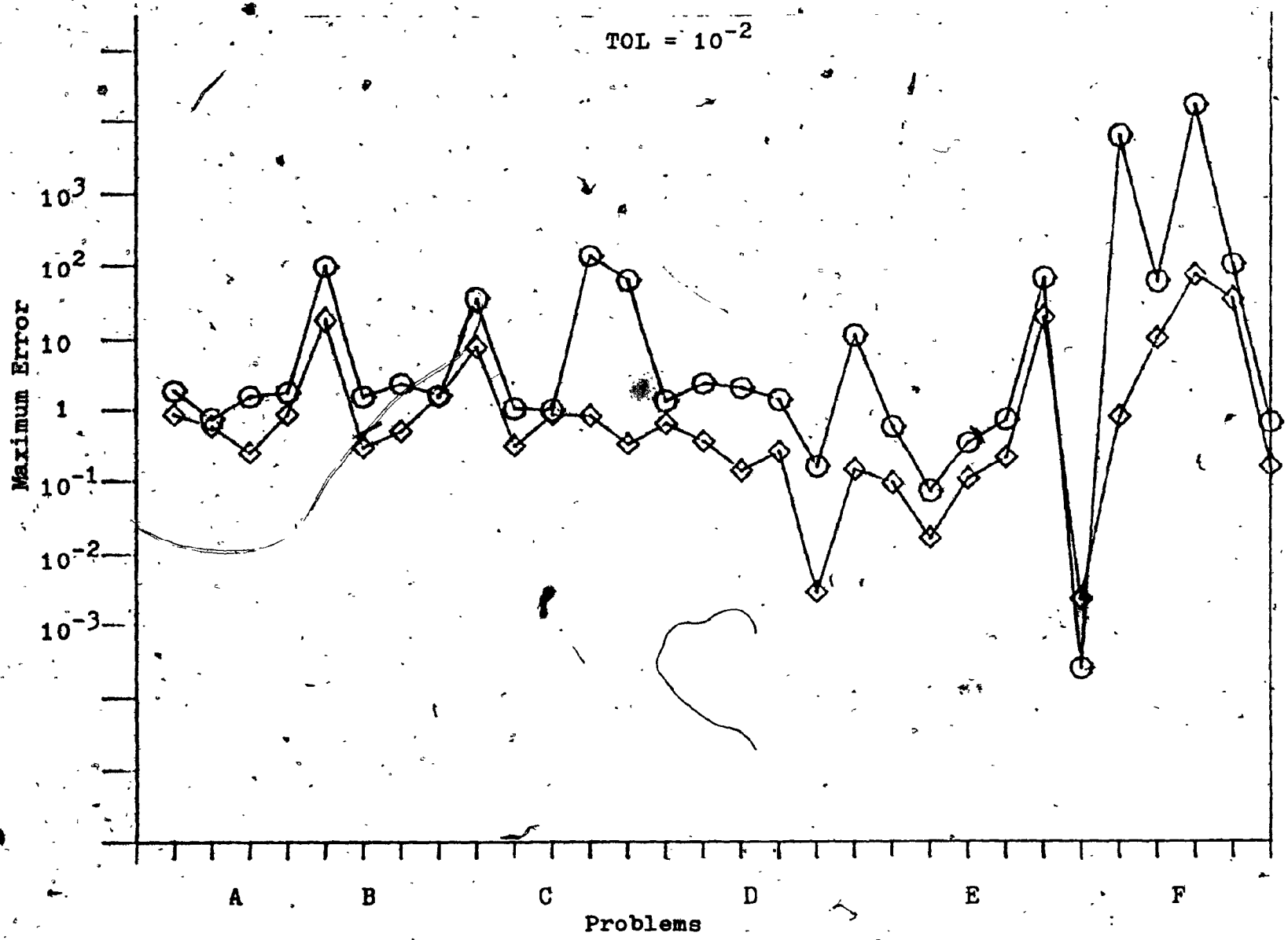
Class B
Linear with complex eigenvalues

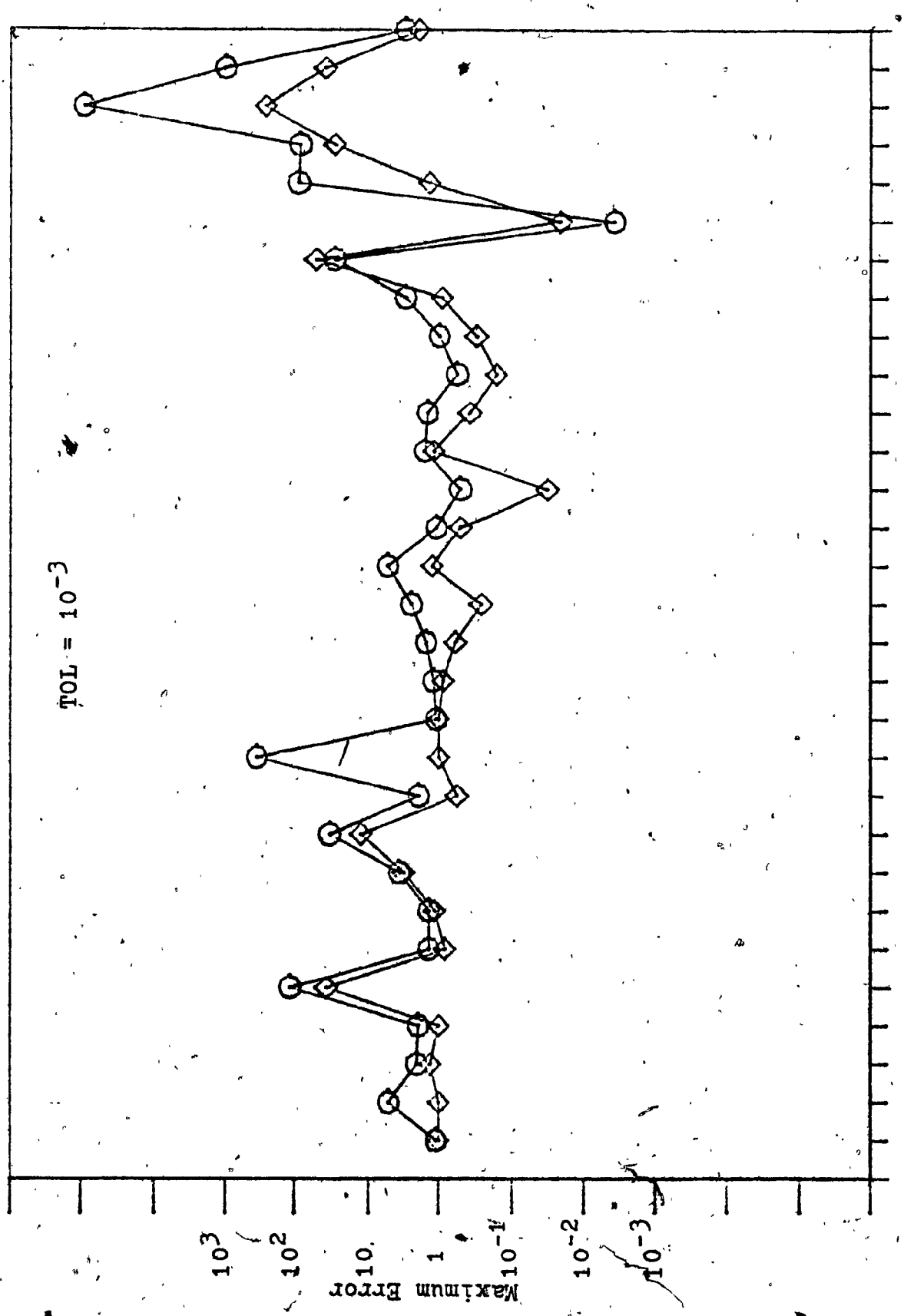


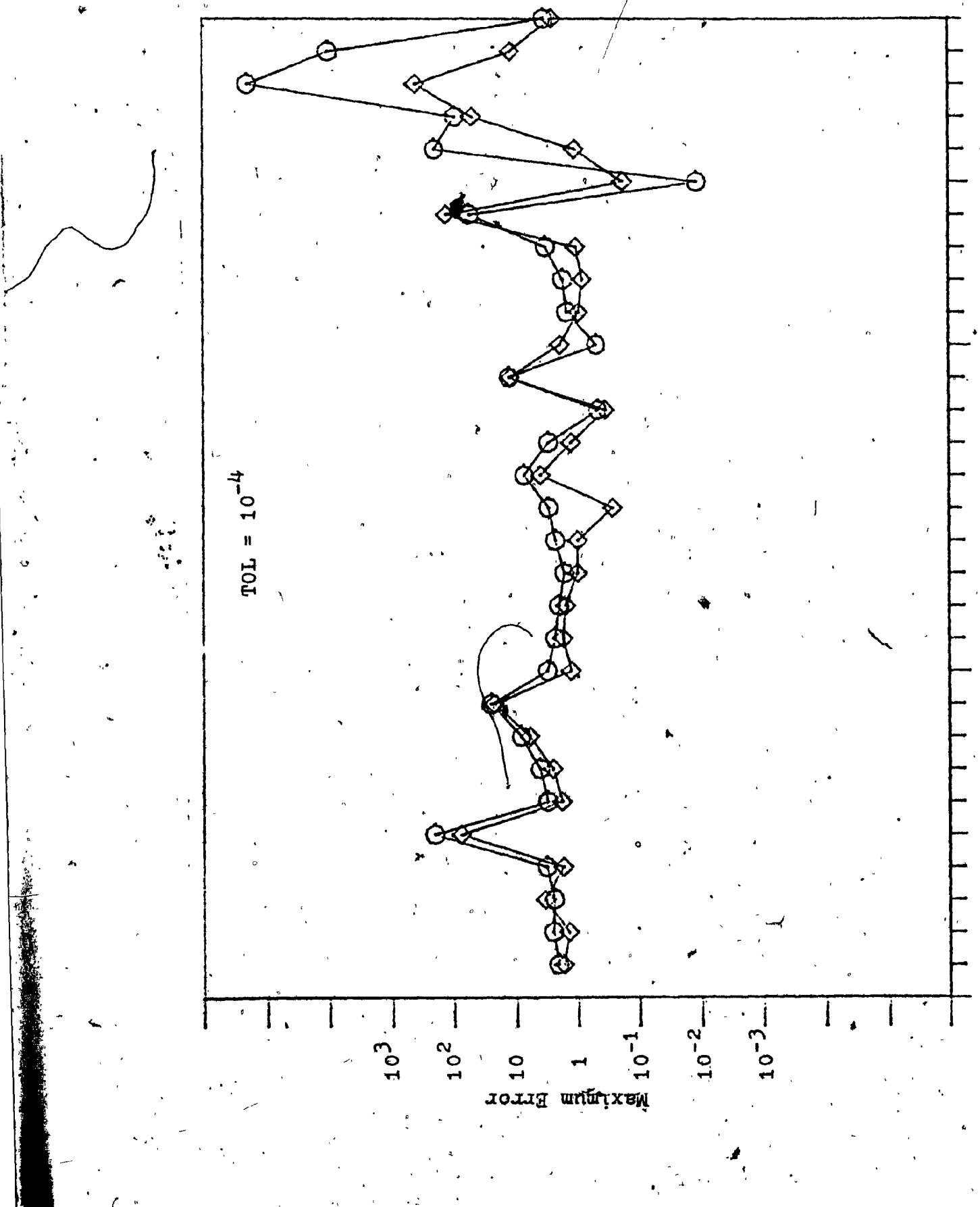
Class F

ROB3A

In the following figure, the maximal global errors are plotted for ROB3A and GEAR for three tolerance TOL = 10^{-2}, 10^{-3}, 10^{-4}.


TOL = 10^{-2}
Maximum Error
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems


TOL = 10^{-3}
Maximum Error
10^{3}
10^{2}
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems


TOL = 10^{-4}
Maximum Error
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems

CLASS A ROWR4(FORMULA 3.14)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	A1	.098	96	16	24	152	12
	A2	.413	132	22	33	319	18
	A3	.168	144	24	36	228	20
	A4	.625	168	28	42	434	24
	SUMMARY	1.304	540	90	135	1133	74
10**-3	A1	.118	120	20	30	190	16
	A2	.484	156	26	39	377	22
	A3	.247	216	36	54	342	26
	A4	.768	216	36	54	558	32
	SUMMARY	1.617	708	118	177	1467	96
10**-4	A1	.208	192	32	48	304	24
	A2	.712	228	38	57	551	30
	A3	.333	288	48	72	456	36
	A4	1.161	312	52	78	806	44
	SUMMARY	2.414	1020	170	255	2117	134

CLASS B ROWR4(FORMULA 3.14)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	B1	.479	408	68	102	646	46
	B2	.167	96	16	24	184	12
	B3	.167	96	16	24	184	12
	B4	.209	120	20	30	230	16
	B5	.390	228	38	57	437	34
	SUMMARY	1.412	948	158	237	1681	120
10**-3	B1	.906	840	140	210	1330	86
	B2	.238	144	24	36	276	16
	B3	.259	156	26	39	299	18
	B4	.322	192	32	48	368	24
	B5	.676	396	66	99	759	58
	SUMMARY	2.401	1728	288	432	3032	202
10**-4	B1	1.669	1512	252	378	2394	150
	B2	.303	180	30	45	345	24
	B3	.331	192	32	48	368	26
	B4	.458	264	44	66	506	38
	B5	1.186	684	114	171	1311	98

	SUMMARY	3.947	2832	472	708	4924	336

CLASS C ROWR4(FORMULA 3.14)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.141	120	20	30	190	16
	C2	.124	108	18	27	171	14
	C3	.133	108	18	27	171	14
	C4	.159	132	22	33	209	18
	C5	.166	132	22	33	209	18
	SUMMARY	.723	600	100	150	950	80
10**-3	C1	.162	144	24	36	228	20
	C2	.176	144	24	36	228	20
	C3	.168	144	24	36	228	20
	C4	.206	168	28	42	266	24
	C5	.234	192	32	48	304	28
	SUMMARY	.946	792	132	198	1254	112
10**-4	C1	.266	228	38	57	361	30
	C2	.251	216	36	54	342	28
	C3	.256	216	36	54	342	28
	C4	.313	252	42	63	399	38
	C5	.378	300	50	75	475	46
	SUMMARY	1.464	1212	202	303	1919	170

CLASS D ROWR4(FORMULA 3.14)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.176	204	34	51	289	12
	D2	.125	144	24	36	204	18
	D3	.154	132	22	33	209	18
	D4	.056	72	12	18	102	8
	D5	.069	120	20	30	150	12
	D6	.073	84	14	21	119	10
	SUMMARY	.653	756	126	189	1073	78
10**-3	D1	.179	216	36	54	306	20
	D2	.143	168	28	42	238	22
	D3	.190	168	28	42	266	24
	D4	.060	72	12	18	102	8
	D5	.069	120	20	30	150	12
	D6	.076	96	16	24	136	12

	SUMMARY	.717	840	140	210	1198	98
10**-4	D1	.284	336	56	84	476	32
	D2	.183	216	36	54	306	30
	D3	.248	216	36	54	342	32
	D4	.076	96	16	24	136	10
	D5	.125	204	34	51	255	18
	D6	.101	120	20	30	170	16
	SUMMARY	1.017	1188	198	297	1685	138

CLASS E ROWR4(FORMULA 3.14)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.083	60	10	15	95	6
	E2	.026	48	8	12	60	4
	E3	.107	132	22	33	187	16
	E4	.424	324	54	81	513	36
	E5	.078	72	12	18	114	6
	SUMMARY	.718	636	106	159	969	68
10**-3	E1	.098	72	12	18	114	8
	E2	.061	108	18	27	135	6
	E3	.120	144	24	36	204	20
	E4	.447	348	58	87	551	44
	E5	.077	72	12	18	114	6
	SUMMARY	.803	744	124	186	1118	84
10**-4	E1	.113	84	14	21	133	10
	E2	.062	108	18	27	135	8
	E3	.203	240	40	60	340	30
	E4	.739	576	96	144	912	64
	E5	.076	72	12	18	114	6
	SUMMARY	1.193	1080	180	270	1634	118

CLASS F ROWR4(FORMULA 3.14)

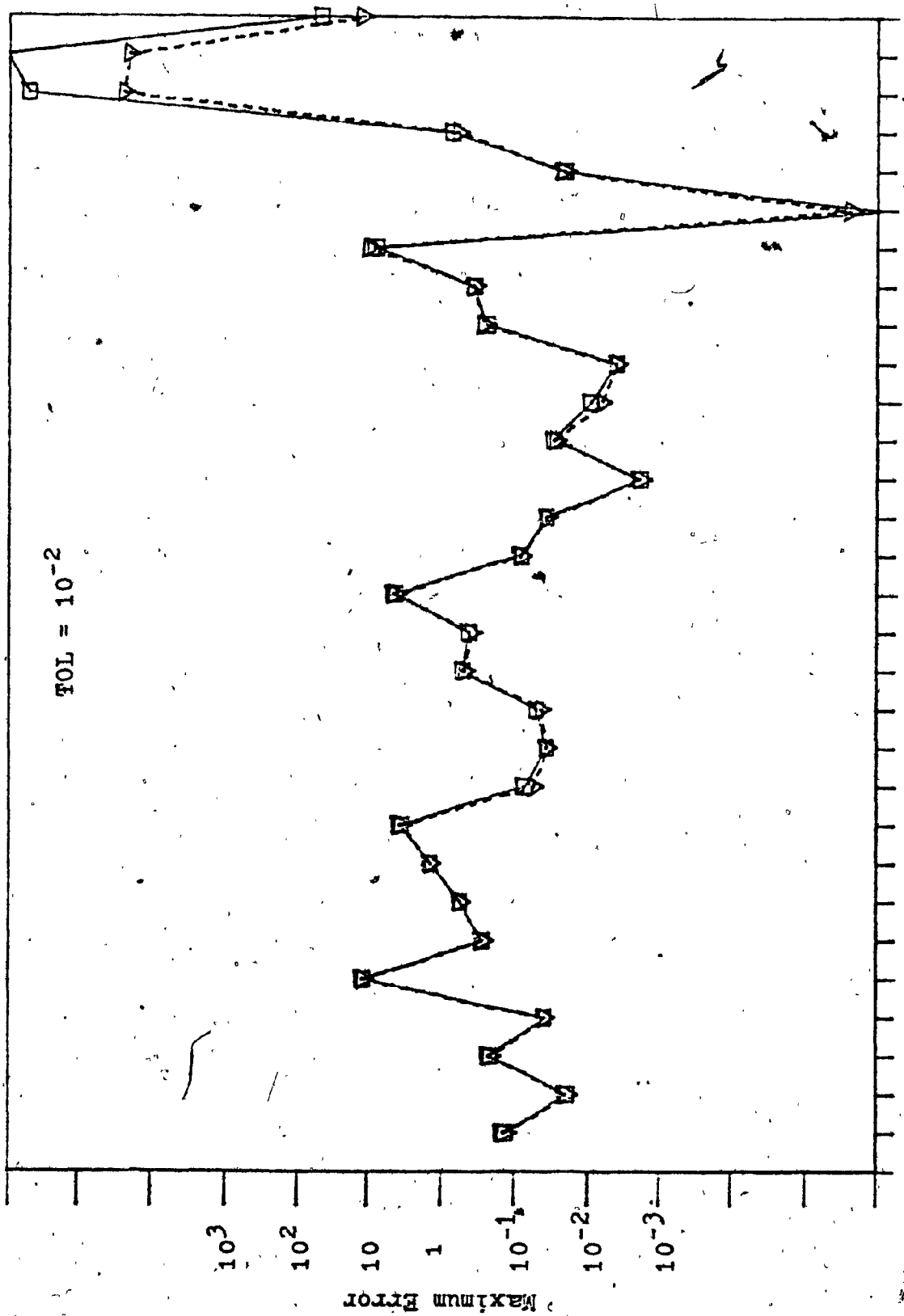
TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.067	120	20	30	150	12
	F2	.283	480	80	120	600	54
	F3	1.316	1500	250	375	2125	134
	F4	.660	768	128	192	1088	70
	F5	.567	204	34	51	459	22

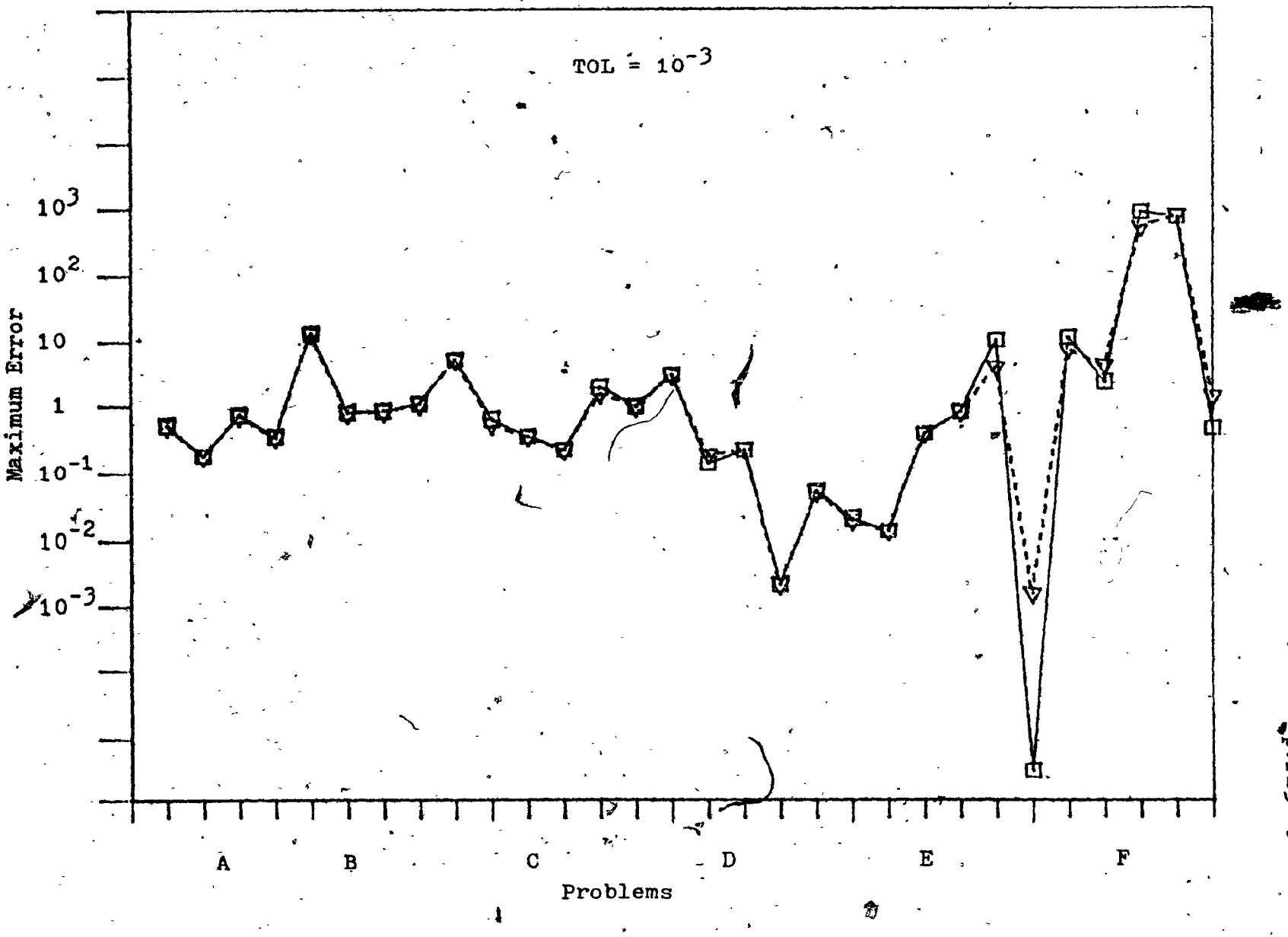
	SUMMARY	2.893	3072	512	768	4422	292
10**-3	F1	.102	180	30	45	225	18
	F2	.179	312	52	78	390	28
	F3	.999	1164	194	291	1649	122
	F4	.676	780	130	195	1105	78
	F5	.670	240	40	60	540	26
	SUMMARY	2.626	2676	446	669	3909	272
10**-4	F1	.196	336	56	84	420	30
	F2	.202	348	58	87	435	36
	F3	1.808	2100	350	525	2975	190
	F4	.869	996	166	249	1411	106
	F5	1.084	384	64	96	864	40
	SUMMARY	4.159	4164	694	1041	6105	402

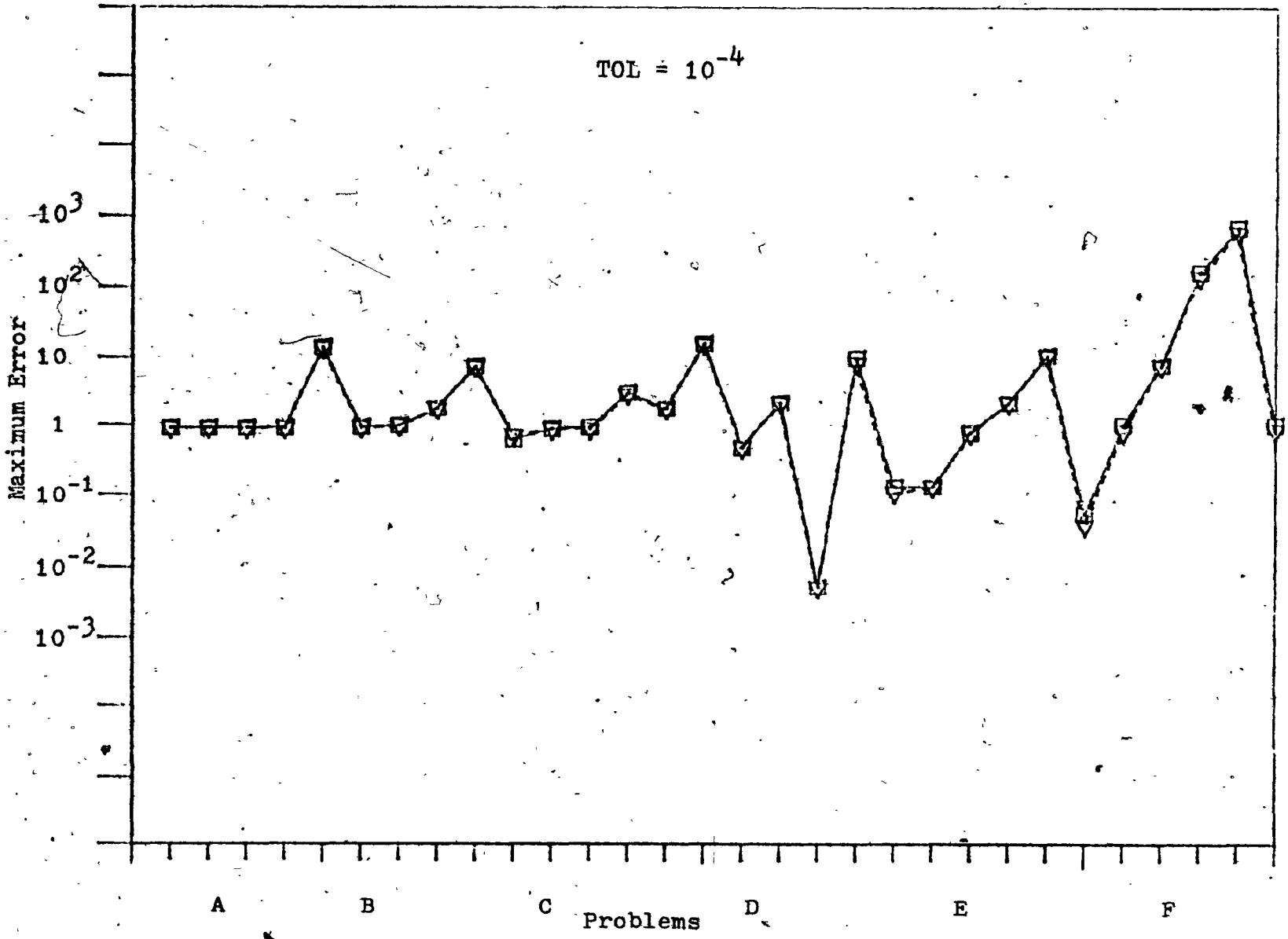
In the following figure, the comparison of the maximal global errors for ROWR4 methods with a3=0.83(formula 3.13) and a3=1.00(formula 3.14) for three tolerance TOL = 10^{-2}, 10^{-3}, 10^{-4} are given.

—▽— ROWR4(formula 3.14)

—□— ROWR4(formula 3.13)


TOL = 10^{-2}
Maximum Error
10^{3}
10^{2}
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems


TOL = 10^{-3}
Maximum Error
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems


TOL = 10^{-4}
Maximum Error
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems

CLASS A. ROWR4(FORMULA 3.15)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	A1	.097	96	16	24	152	12
	A2	.416	132	22	33	319	18
	A3	.254	216	36	54	342	18
	A4	.614	168	28	42	434	24
	SUMMARY	1.381	612	102	153	1247	72
10**-3	A1	.166	156	26	39	247	16
	A2	.492	156	26	39	377	22
	A3	.290	252	42	63	399	26
	A4	.735	204	34	51	527	30
	SUMMARY	1.683	768	128	192	1550	94
10**-4	A1	.209	192	32	48	304	24
	A2	.674	216	36	54	522	30
	A3	.323	276	46	69	437	34
	A4	1.049	288	48	72	744	42
	SUMMARY	2.255	972	162	243	2007	130

CLASS B ROWR4(FORMULA 3.15)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	B1	.458	408	68	102	646	48
	B2	.264	156	26	39	299	10
	B3	.248	144	24	36	276	12
	B4	.252	144	24	36	276	16
	B5	.396	228	38	57	437	34
	SUMMARY	1.618	1080	180	270	1934	120
10**-3	B1	.974	900	150	225	1425	90
	B2	.259	156	26	39	299	16
	B3	.278	168	28	42	322	18
	B4	.302	180	30	45	345	24
	B5	.651	384	64	96	736	58
	SUMMARY	2.464	1788	298	447	3127	206
10**-4	B1	1.580	1428	238	357	2261	142
	B2	.308	180	30	45	345	22
	B3	.313	180	30	45	345	24
	B4	.443	252	42	63	483	36
	B5	1.221	684	114	171	1311	98

	SUMMARY	3.865	2724	454	681	4745	322

CLASS C ROWR4(FORMULA 3.15)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.135	120	20	30	190	16
	C2	.128	108	18	27	171	14
	C3	.129	108	18	27	171	14
	C4	.167	132	22	33	209	18
	C5	.162	132	22	33	209	18
	SUMMARY	.721	600	100	150	950	80
10**-3	C1	.161	144	24	36	228	20
	C2	.176	144	24	36	228	20
	C3	.169	144	24	36	228	20
	C4	.223	180	30	45	285	26
	C5	.234	192	32	48	304	28
	SUMMARY	.963	804	134	201	1273	114
10**-4	C1	.233	204	34	51	323	28
	C2	.237	204	34	51	323	28
	C3	.271	228	38	57	361	32
	C4	.509	408	68	102	646	48
	C5	.499	396	66	99	627	58
	SUMMARY	1.749	1440	240	360	2280	194

CLASS D ROWR4(FORMULA 3.15)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.236	276	46	69	391	14
	D2	.132	156	26	39	221	18
	D3	.167	144	24	36	228	20
	D4	.057	72	12	18	102	8
	D5	.056	96	16	24	120	12
	D6	.081	96	16	24	136	12
	SUMMARY	.729	840	140	210	1198	84
10**-3	D1	.273	324	54	81	459	24
	D2	.152	180	30	45	255	24
	D3	.204	180	30	45	285	26
	D4	.057	72	12	18	102	8
	D5	.077	132	22	33	165	14
	D6	.088	108	18	27	153	14

	SUMMARY	.851	996	166	249	1419	110
10**-4	D1	.423	504	84	126	714	40
	D2	.220	252	42	63	357	36
	D3	.313	276	46	69	437	36
	D4	.066	84	14	21	119	10
	D5	.163	264	44	66	330	18
	D6	.107	132	22	33	187	18
	SUMMARY	1.292	1512	252	378	2144	158

CLASS E ROWR4(FORMULA 3.15)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.078	60	10	15	95	6
	E2	.019	36	6	9	45	2
	E3	.099	120	20	30	170	16
	E4	.385	300	50	75	475	36
	E5	.094	84	14	21	133	6
	SUMMARY	.675	600	100	150	918	66
10**-3	E1	.097	72	12	18	114	8
	E2	.075	132	22	33	165	6
	E3	.141	168	28	42	238	24
	E4	.463	360	60	90	570	46
	E5	.091	84	14	21	133	6
	SUMMARY	.867	816	136	204	1220	90
10**-4	E1	.115	84	14	21	133	10
	E2	.061	108	18	27	135	8
	E3	.242	288	48	72	408	38
	E4	.656	516	86	129	817	64
	E5	.078	72	12	18	114	6
	SUMMARY	1.152	1068	178	267	1607	126

CLASS F ROWR4(FORMULA 3.15)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.141	228	38	57	285	28
	F2	.541	900	150	225	1125	78
	F3	1.050	1188	198	297	1683	106
	F4	.779	912	152	228	1292	76
	F5	.360	132	22	33	297	16

	SUMMARY	2.871	3360	560	840	4682	304
10**-3	F1	.104	180	30	45	225	18
	F2	.244	420	70	105	525	44
	F3	1.404	1632	272	408	2312	160
	F4	.747	864	144	216	1224	88
	F5	.900	324	54	81	729	34
	SUMMARY	3.399	3420	570	855	5015	344
10**-4	F1	.181	312	52	78	390	24
	F2	.426	720	120	180	900	84
	F3	1.805	2100	350	525	2975	214
	F4	1.026	1176	196	294	1666	124
	F5	.870	312	52	78	702	36
	SUMMARY	4.308	4620	770	1155	6633	482

CLASS A ROWR4(FORMULA 3.16)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	A1	.115	108	18	27	171	14
	A2	.418	132	22	33	319	18
	A3	.163	144	24	36	228	20
	A4	.608	168	28	42	434	24
	SUMMARY	1.304	552	92	138	1152	76
10**-3	A1	.172	156	26	39	247	18
	A2	.522	168	28	42	406	24
	A3	.240	204	34	51	323	30
	A4	.963	264	44	66	682	36
	SUMMARY	1.897	792	132	198	1658	108
10**-4	A1	.257	252	42	63	399	30
	A2	.869	276	46	69	667	36
	A3	.373	324	54	81	513	44
	A4	1.424	396	66	99	1023	56
	SUMMARY	2.923	1248	208	312	2602	166

CLASS B ROWR4(FORMULA 3.16)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	B1	.562	516	86	129	817	52
	B2	.160	96	16	24	184	12
	B3	.187	108	18	27	207	14
	B4	.226	132	22	33	253	18
	B5	.360	204	34	51	391	30
	SUMMARY	1.495	1056	176	264	1852	126
10**-3	B1	1.087	1008	168	252	1596	110
	B2	.245	144	24	36	276	20
	B3	.239	144	24	36	276	20
	B4	.342	204	34	51	391	30
	B5	.838	480	80	120	920	68
	SUMMARY	2.751	1980	330	495	3459	248
10**-4	B1	2.241	2028	338	507	3211	208
	B2	.376	216	36	54	414	30
	B3	.390	228	38	57	437	32
	B4	.598	336	56	84	644	50
	B5	1.812	1020	170	255	1955	134

	SUMMARY	5.417	3828	638	957	6661	454

CLASS C ROWR4(FORMULA 3.16)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.140	120	20	30	190	16
	C2	.142	120	20	30	190	16
	C3	.153	120	20	30	190	16
	C4	.179	144	24	36	228	20
	C5	.179	144	24	36	228	20
	SUMMARY	.793	648	108	162	1026	88
10**-3	C1	.232	204	34	51	323	24
	C2	.210	180	30	45	285	22
	C3	.182	156	26	39	247	22
	C4	.284	228	38	57	361	30
	C5	.263	216	36	54	342	32
	SUMMARY	1.171	984	164	246	1558	130
10**-4	C1	.316	276	46	69	437	38
	C2	.308	264	44	66	418	34
	C3	.330	276	46	69	437	36
	C4	.457	372	62	93	589	48
	C5	.539	432	72	108	684	56
	SUMMARY	1.950	1620	270	405	2565	212

CLASS D ROWR4(FORMULA 3.16)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.234	276	46	69	391	14
	D2	.139	156	26	39	221	18
	D3	.170	144	24	36	228	20
	D4	.048	60	10	15	85	6
	D5	.041	72	12	18	90	8
	D6	.069	84	14	21	119	10
	SUMMARY	.701	792	132	198	1134	76
10**-3	D1	.333	396	66	99	561	26
	D2	.153	180	30	45	255	24
	D3	.207	180	30	45	285	26
	D4	.046	60	10	15	85	6
	D5	.054	96	16	24	120	10
	D6	.077	96	16	24	136	12

	SUMMARY	.870	1008	168	252	1442	104
10**-4	D1	.540	624	104	156	884	48
	D2	.206	240	40	60	340	34
	D3	.331	288	48	72	456	38
	D4	.067	84	14	21	119	10
	D5	.102	156	26	39	195	16
	D6	.100	120	20	30	170	16
	SUMMARY	1.346	1512	252	378	2164	162

CLASS E ROWR4(FORMULA 3.16)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.078	60	10	15	95	6
	E2	.026	48	8	12	60	4
	E3	.100	120	20	30	170	16
	E4	.393	300	50	75	475	36
	E5	.162	144	24	36	228	10
	SUMMARY	.759	672	112	168	1028	72
10**-3	E1	.097	72	12	18	114	8
	E2	.033	60	10	15	75	6
	E3	.141	168	28	42	238	24
	E4	.537	420	70	105	665	54
	E5	.077	72	12	18	114	6
	SUMMARY	.885	792	132	198	1206	98
10**-4	E1	.112	84	14	21	133	10
	E2	.067	120	20	30	150	10
	E3	.310	372	62	93	527	40
	E4	.923	720	120	180	1140	82
	E5	.091	84	14	21	133	8
	SUMMARY	1.503	1380	230	345	2083	150

CLASS F ROWR4(FORMULA 3.16)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.057	96	16	24	120	8
	F2	.177	300	50	75	375	30
	F3	1.336	1488	248	372	2108	132
	F4	.700	804	134	201	1139	70
	F5	.571	204	34	51	459	22

	SUMMARY	2.841	2892	482	723	4201	262
10**-3	F1	.160	276	46	69	345	32
	F2	.217	372	62	93	465	38
	F3	1.800	2064	344	516	2924	186
	F4	.854	984	164	246	1394	90
	F5	1.002	360	60	90	810	42
	SUMMARY	4.033	4056	676	1014	5938	388
10**-4	F1	.146	252	42	63	315	24
	F2	.274	468	78	117	585	48
	F3	2.318	2688	448	672	3808	246
	F4	1.178	1344	224	336	1904	134
	F5	.976	348	58	87	783	42
	SUMMARY	4.892	5100	850	1275	7395	494

CLASS A ROWR4(FORMULA 3.17)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	A1	.111	108	18	27	171	14
	A2	.420	132	22	33	319	18
	A3	.189	156	26	39	247	22
	A4	.619	168	28	42	434	24
	SUMMARY	1.339	564	94	141	1171	78
10**-3	A1	.195	180	30	45	285	20
	A2	.532	168	28	42	406	24
	A3	.234	204	34	51	323	30
	A4	.997	276	46	69	713	36
	SUMMARY	1.958	828	138	207	1727	110
10**-4	A1	.258	252	42	63	399	32
	A2	.900	288	48	72	696	38
	A3	.356	312	52	78	494	46
	A4	1.494	408	68	102	1054	58
	SUMMARY	3.008	1260	210	315	2643	174

CLASS B ROWR4(FORMULA 3.17)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	B1	.588	528	88	132	836	54
	B2	.190	108	18	27	207	14
	B3	.187	108	18	27	207	14
	B4	.224	132	22	33	253	18
	B5	.358	204	34	51	391	30
	SUMMARY	1.547	1080	180	270	1894	130
10**-3	B1	1.183	1092	182	273	1729	106
	B2	.240	144	24	36	276	20
	B3	.258	156	26	39	299	22
	B4	.344	204	34	51	391	30
	B5	.842	492	82	123	943	68
	SUMMARY	2.867	2088	348	522	3638	246
10**-4	B1	2.337	2100	350	525	3325	216
	B2	.371	216	36	54	414	30
	B3	.421	240	40	60	460	34
	B4	.604	336	56	84	644	50
	B5	1.778	1020	170	255	1955	138

	SUMMARY	5.511	3912	652	978	6798	468

CLASS C ROWR4(FORMULA 3.17)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.133	120	20	30	190	16
	C2	.139	120	20	30	190	16
	C3	.145	120	20	30	190	16
	C4	.180	144	24	36	228	20
	C5	.191	156	26	39	247	22
	SUMMARY	.788	660	110	165	1045	90
10**-3	C1	.232	204	34	51	323	24
	C2	.225	192	32	48	304	22
	C3	.187	156	26	39	247	22
	C4	.281	228	38	57	361	30
	C5	.279	228	38	57	361	34
	SUMMARY	1.204	1008	168	252	1596	132
10**-4	C1	.325	288	48	72	456	38
	C2	.321	276	46	69	437	36
	C3	.326	276	46	69	437	36
	C4	.526	420	70	105	665	50
	C5	.660	528	88	132	836	60
	SUMMARY	2.158	1788	298	447	2831	220

CLASS D ROWR4(FORMULA 3.17)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.230	276	46	69	391	14
	D2	.127	144	24	36	204	18
	D3	.172	144	24	36	228	20
	D4	.046	60	10	15	85	6
	D5	.049	84	14	21	105	10
	D6	.068	84	14	21	119	10
	SUMMARY	.692	792	132	198	1132	78
10**-3	D1	.371	444	74	111	629	28
	D2	.154	180	30	45	255	24
	D3	.215	192	32	48	304	28
	D4	.056	72	12	18	102	8
	D5	.048	84	14	21	105	10
	D6	.077	96	16	24	136	12

	SUMMARY	.921	1068	178	267	1531	110
10**-4	D1	.529	624	104	156	884	50
	D2	.206	240	40	60	340	34
	D3	.357	312	52	78	494	40
	D4	.065	84	14	21	119	10
	D5	.124	204	34	51	255	20
	D6	.099	120	20	30	170	16
	SUMMARY	1.380	1584	264	396	2262	170

CLASS E ROWR4(FORMULA 3.17)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.081	60	10	15	95	6
	E2	.025	48	8	12	60	4
	E3	.104	120	20	30	170	16
	E4	.367	288	48	72	456	38
	E5	.106	96	16	24	152	6
	SUMMARY	.683	612	102	153	933	70
10**-3	E1	.096	72	12	18	114	8
	E2	.033	60	10	15	75	6
	E3	.138	168	28	42	238	24
	E4	.633	492	82	123	779	54
	E5	.089	84	14	21	133	6
	SUMMARY	.989	876	146	219	1339	98
10**-4	E1	.113	84	14	21	133	10
	E2	.068	120	20	30	150	10
	E3	.300	360	60	90	510	40
	E4	.896	696	116	174	1102	84
	E5	.092	84	14	21	133	8
	SUMMARY	1.469	1344	224	336	2028	152

CLASS F ROWR4(FORMULA 3.17)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.057	96	16	24	120	8
	F2	.154	252	42	63	315	26
	F3	1.287	1452	242	363	2057	122
	F4	.872	1020	170	255	1445	86
	F5	.398	144	24	36	324	16

	SUMMARY	2.768	2964	494	741	4261	258
10**-3	F1	.137	240	40	60	300	28
	F2	.417	720	120	180	900	82
	F3	1.602	1848	308	462	2618	166
	F4	.783	924	154	231	1309	90
	F5	.702	252	42	63	567	28
	SUMMARY	3.641	3984	664	996	5694	394
10**-4	F1	.147	252	42	63	315	24
	F2	.282	480	80	120	600	50
	F3	2.436	2784	464	696	3944	252
	F4	1.160	1356	226	339	1921	136
	F5	.975	348	58	87	783	42
	SUMMARY	5.000	5220	870	1305	7563	504

CLASS A ROWR4(FORMULA 3.18)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	A1	.098	96	16	24	152	12
	A2	.397	120	20	30	290	16
	A3	.151	132	22	33	209	18
	A4	.585	156	26	39	403	22
	SUMMARY	1.231	504	84	126	1054	68
10**-3	A1	.121	120	20	30	190	16
	A2	.447	144	24	36	348	20
	A3	.190	168	28	42	266	24
	A4	.725	204	34	51	527	30
	SUMMARY	1.483	636	106	159	1331	90
10**-4	A1	.197	192	32	48	304	24
	A2	.679	216	36	54	522	28
	A3	.312	264	44	66	418	34
	A4	1.099	300	50	75	775	42
	SUMMARY	2.287	972	162	243	2019	128

CLASS B ROWR4(FORMULA 3.18)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	B1	.512	456	76	114	722	50
	B2	.140	84	14	21	161	10
	B3	.162	96	16	24	184	12
	B4	.201	120	20	30	230	16
	B5	.370	216	36	54	414	32
	SUMMARY	1.385	972	162	243	1711	120
10**-3	B1	.871	804	134	201	1273	80
	B2	.241	144	24	36	276	16
	B3	.260	156	26	39	299	16
	B4	.333	192	32	48	368	24
	B5	.605	360	60	90	690	54
	SUMMARY	2.310	1656	276	414	2906	190
10**-4	B1	1.427	1284	214	321	2033	138
	B2	.307	180	30	45	345	22
	B3	.306	180	30	45	345	24
	B4	.437	252	42	63	483	36
	B5	1.063	612	102	153	1173	92

	SUMMARY	3.540	2508	418	627	4379	312

CLASS C ROWR4(FORMULA 3.18)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	C1	.122	108	18	27	171	14
	C2	.125	108	18	27	171	14
	C3	.127	108	18	27	171	14
	C4	.147	120	20	30	190	16
	C5	.167	132	22	33	209	18
	SUMMARY	.688	576	96	144	912	76
10**-3	C1	.176	156	26	39	247	20
	C2	.152	132	22	33	209	18
	C3	.169	144	24	36	228	20
	C4	.207	168	28	42	266	24
	C5	.202	168	28	42	266	24
	SUMMARY	.906	768	128	192	1216	106
10**-4	C1	.234	204	34	51	323	28
	C2	.238	204	34	51	323	26
	C3	.256	216	36	54	342	28
	C4	.284	228	38	57	361	32
	C5	.321	264	44	66	418	34
	SUMMARY	1.333	1116	186	279	1767	148

CLASS D ROWR4(FORMULA 3.18)

TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	D1	.182	216	36	54	306	12
	D2	.138	156	26	39	221	18
	D3	.152	132	22	33	209	18
	D4	.047	60	10	15	85	6
	D5	.064	108	18	27	135	10
	D6	.074	84	14	21	119	10
	SUMMARY	.657	756	126	189	1075	74
10**-3	D1	.179	216	36	54	306	20
	D2	.151	180	30	45	255	24
	D3	.204	180	30	45	285	26
	D4	.057	72	12	18	102	8
	D5	.054	96	16	24	120	10
	D6	.079	96	16	24	136	12

	SUMMARY	.724	840	140	210	1204	100
10**-4	D1	.425	504	84	126	714	34
	D2	.195	228	38	57	323	32
	D3	.300	264	44	66	418	34
	D4	.067	84	14	21	119	8
	D5	.132	216	36	54	270	20
	D6	.098	120	20	30	170	14
	SUMMARY	1.217	1416	236	354	2014	142

CLASS E ROWR4(FORMULA 3.18)

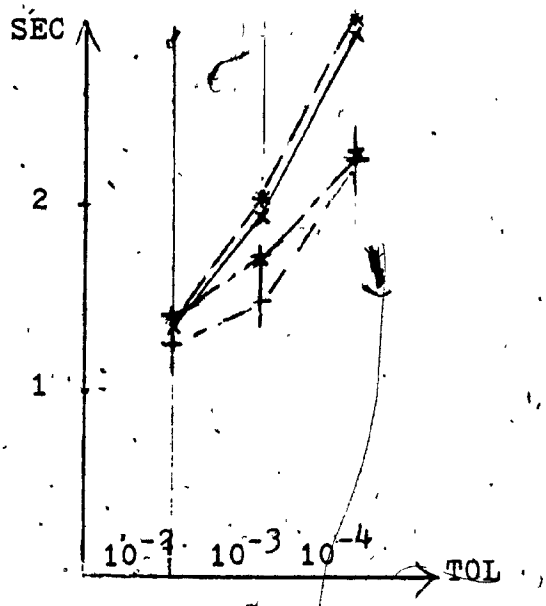
TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	E1	.079	60	10	15	95	6
	E2	.026	48	8	12	60	4
	E3	.128	144	24	36	204	16
	E4	.381	300	50	75	475	34
	E5	.098	84	14	21	133	6
	SUMMARY	.712	636	106	159	967	66
10**-3	E1	.095	72	12	18	114	8
	E2	.041	72	12	18	90	4
	E3	.137	168	28	42	238	22
	E4	.476	372	62	93	589	46
	E5	.077	72	12	18	114	6
	SUMMARY	.826	756	126	189	1145	86
10**-4	E1	.113	84	14	21	133	10
	E2	.067	120	20	30	150	8
	E3	.228	264	44	66	374	34
	E4	.660	516	86	129	817	64
	E5	.078	72	12	18	114	6
	SUMMARY	1.146	1056	176	264	1588	122

CLASS F ROWR4(FORMULA 3.18)

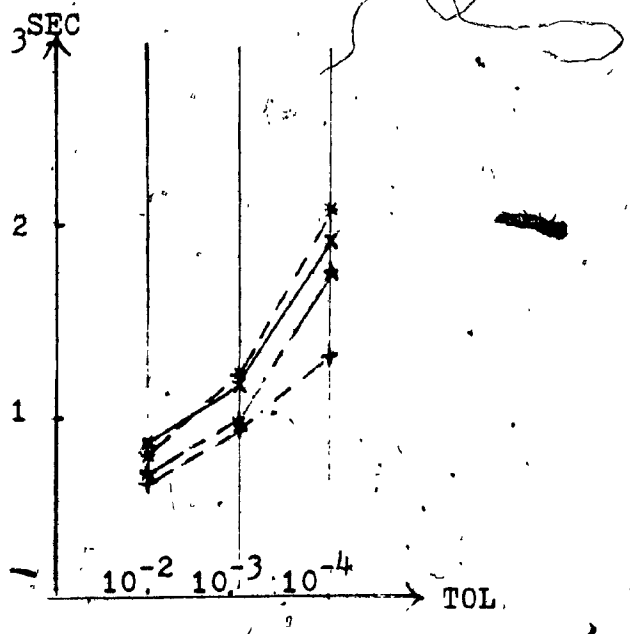
TOL		TIME	FCN CALLS	JAC CALLS	INV CALLS	TF	NO OF STEPS
10**-2	F1	.125	204	34	51	255	22
	F2	.230	384	64	96	480	38
	F3	1.111	1272	212	318	1802	120
	F4	.703	828	138	207	1173	72
	F5	.636	228	38	57	513	18

	SUMMARY	2.805	2916	486	729	4223	270
10**-3	F1	.106	180	30	45	225	16
	F2	.167	288	48	72	360	28
	F3	1.087	1260	210	315	1785	132
	F4	.651	768	128	192	1088	78
	F5	.666	240	40	60	540	28
	SUMMARY	2.677	2736	456	684	3998	282
10**-4	F1	.159	276	46	69	345	26
	F2	.204	348	58	87	435	36
	F3	1.518	1716	286	429	2431	186
	F4	.833	984	164	246	1394	106
	F5	.977	348	58	87	783	38
	SUMMARY	3.691	3672	612	918	5388	392

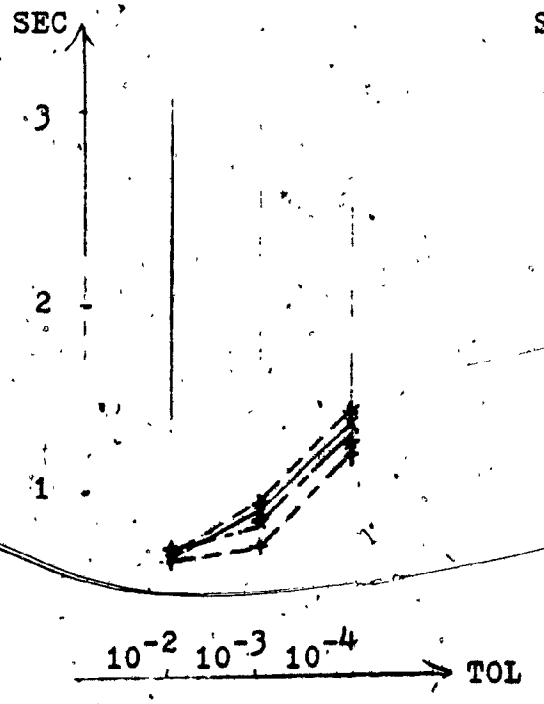
The following graphs show the computing time of ROWR4 with different γ values over the six different classes of problems.



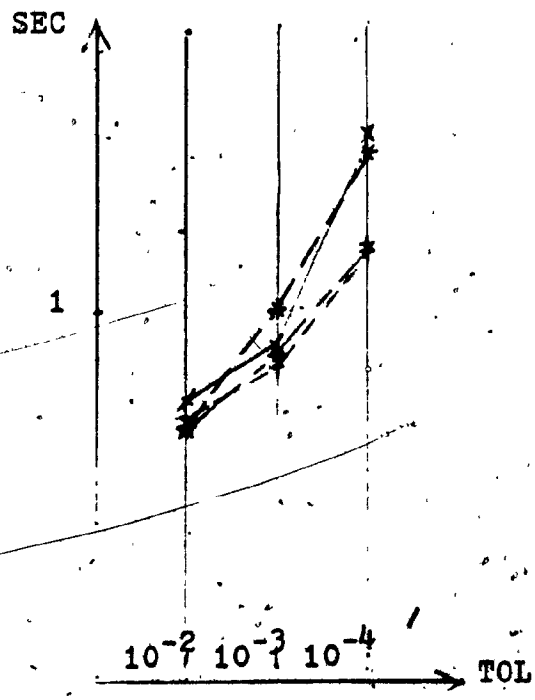
Class A
Linear with real eigenvalues



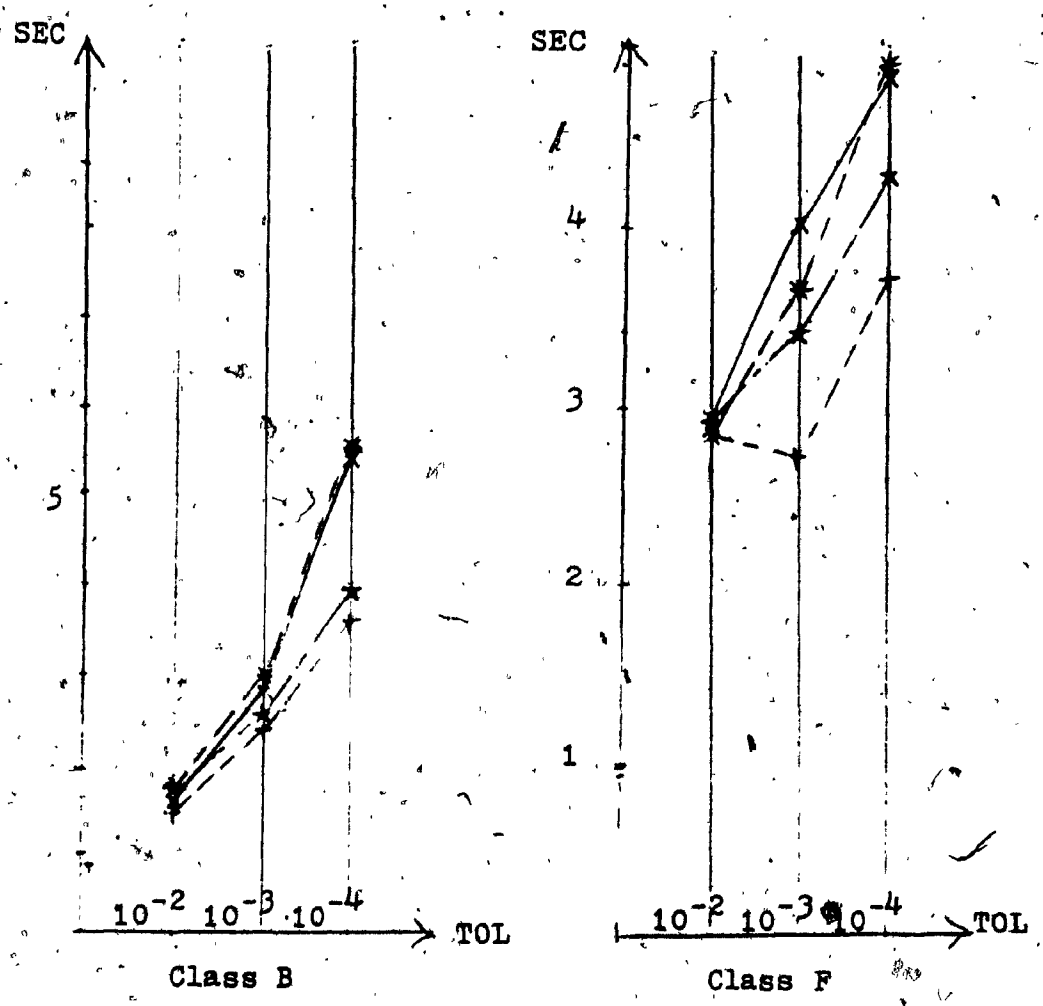
Class C
Non-linear coupling



Class D
Nonlinear with real eigenvalues



Class E
Nonlinear with complex eigenvalues



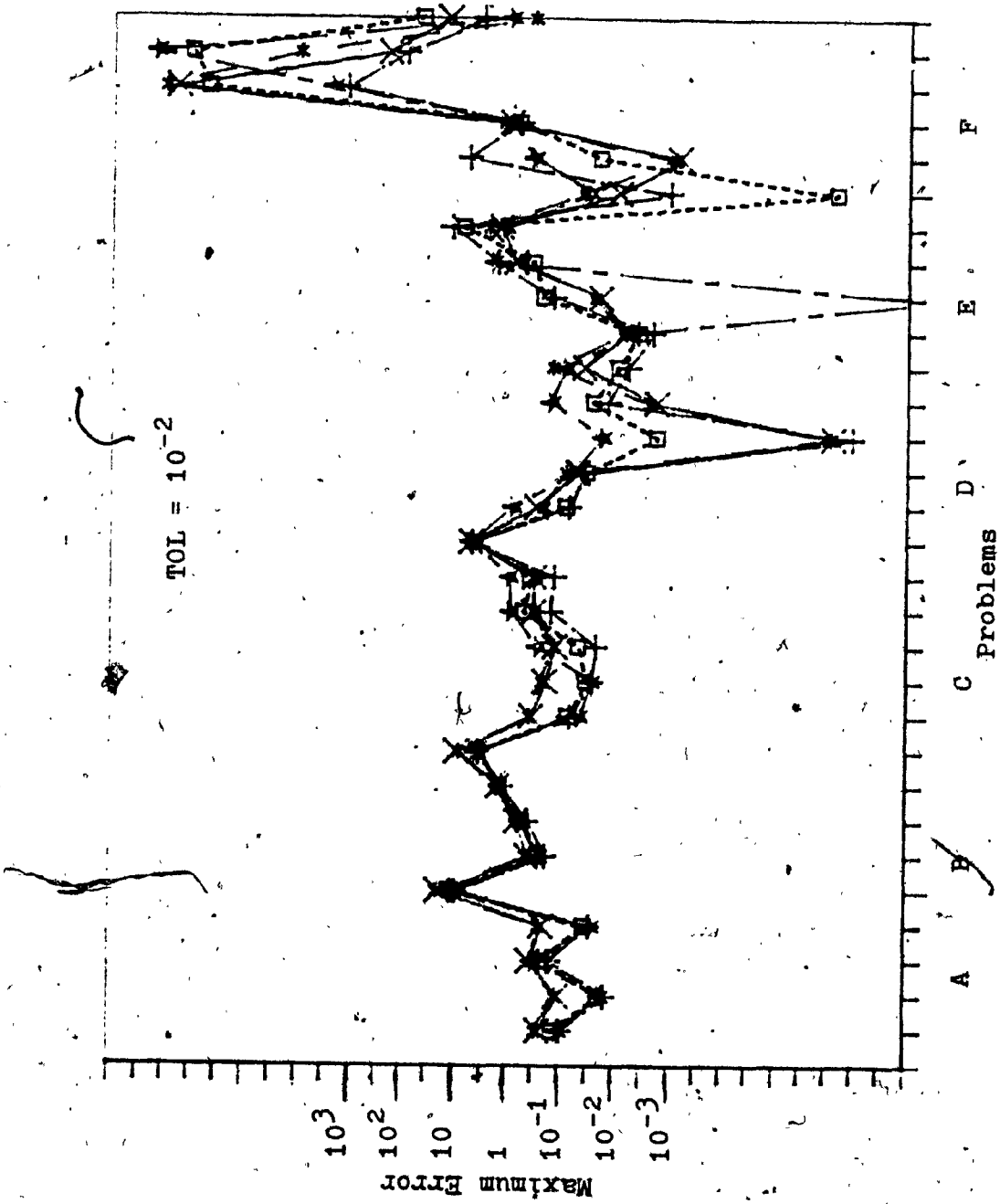
ROWR4(formula 3.15)

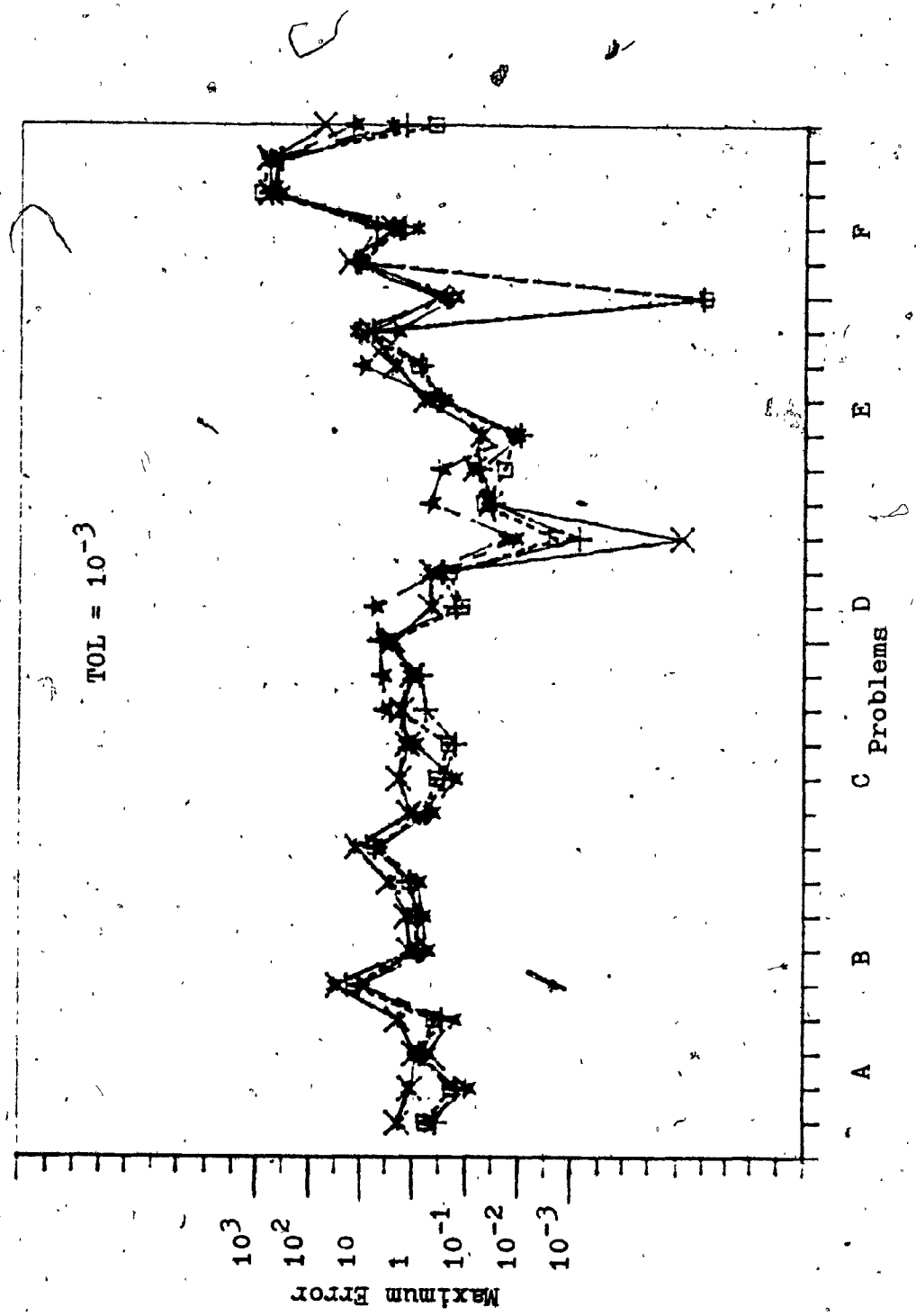
ROWR4(formula 3.16)

ROWR4(formula 3.17)

ROWR4(formula 3.18)

In the following figure, the maximal global errors are plotted for ROWR4 with different γ values for three tolerance TOL=10^{-2}, 10^{-3}, 10^{-4}.


TOL = 10^{-2}
Maximum Error
10^{3}
10^{2}
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems


TOL = 10^{-3}
Maximum Error
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
A
B
C
D
E
F
Problems